



A MINIMUM EFFORT CONTROL APPROACH  
TO GUIDED MUNITION PATH PLANNING

THESIS

Jeffrey M. Borkowski, First Lieutenant, USAF

AFIT/GE/ENG/06-07

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/06-07

A MINIMUM EFFORT CONTROL APPROACH  
TO GUIDED MUNITION PATH PLANNING

THESIS

Presented to the Faculty  
Department of Electrical and Computer Engineering  
Graduate School of Engineering and Management  
Air Force Institute of Technology  
Air University  
Air Education and Training Command  
In Partial Fulfillment of the Requirements for the  
Degree of Master of Science in Electrical Engineering

Jeffrey M. Borkowski, B.S.E.C.E.

First Lieutenant, USAF

March 2006

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

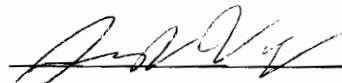
AFIT/GE/ENG/06-07

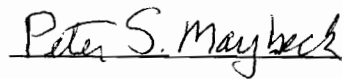
A MINIMUM EFFORT CONTROL APPROACH  
TO GUIDED MUNITION PATH PLANNING

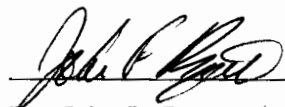
Jeffrey M. Borkowski, B.S.E.C.E.

First Lieutenant, USAF

Approved:

  
Lt. Col. Juan R. Vasquez (Chairman)      Date

  
Dr. Peter S. Maybeck (Member)      Date

  
Dr. John F. Raquet (Member)      Date

*Abstract*

An advance in the development of smart munitions entails autonomously modifying target selection during flight in order to maximize the value of the target being destroyed. Target identification and classification provides a basis for target value which is used in conjunction with multi-target tracks to determine an optimal aimpoint for the munition. A unique guidance law can be constructed that exploits attribute and kinematic data from an onboard video sensor. This thesis develops an innovative path planning algorithm that provides an obstacle avoidance function while navigating the munition toward the highest value target. The foundation of this path planning method is found in the principles of minimum effort control optimization. Results demonstrate the ability of the path planning algorithm to determine a path for the munition to follow which is both stable and feasible.

## *Table of Contents*

	Page
Abstract . . . . .	iii
List of Figures . . . . .	vii
List of Tables . . . . .	ix
List of Abbreviations . . . . .	x
 I. Introduction . . . . .	 1-1
1.1 Research Goals and Contributions . . . . .	1-4
1.2 Scope and Assumptions . . . . .	1-5
1.3 Thesis Organization . . . . .	1-6
 II. Background . . . . .	 2-1
2.1 Introduction . . . . .	2-1
2.2 Target Tracking Basics . . . . .	2-1
2.2.1 Stochastic Estimation and Linear Kalman Filtering	2-1
2.2.2 Target Models . . . . .	2-4
2.3 Control Concepts . . . . .	2-9
2.3.1 Circle-Line-Circle Control . . . . .	2-9
2.3.2 Model Predictive Control . . . . .	2-13
2.3.3 Controllability Matrices and Grammians . . . . .	2-17
2.3.4 Minimum Effort Control . . . . .	2-20
2.4 F-16 Simulation Model . . . . .	2-20
2.5 Summary . . . . .	2-23

	Page
III. Algorithm Development and Simulation Description . . . . .	3-1
3.1 Introduction . . . . .	3-1
3.2 Target Tracker . . . . .	3-1
3.3 Waypoint Generator . . . . .	3-4
3.4 Path Selection . . . . .	3-10
3.5 Command Generator . . . . .	3-18
3.5.1 Extremal Input Commanding . . . . .	3-18
3.5.2 Model Predictive Control Commanding . . . . .	3-20
3.5.3 Minimum Effort Control Commanding . . . . .	3-22
3.6 Flight Simulator . . . . .	3-23
3.7 Summary . . . . .	3-25
IV. Simulation Implementation and Results . . . . .	4-1
4.1 Introduction . . . . .	4-1
4.2 Command Generators . . . . .	4-2
4.2.1 Model Predictive Control Command Generator . . . . .	4-2
4.2.2 MEP-based Command Generator . . . . .	4-6
4.3 Basic Demonstration . . . . .	4-8
4.4 Test Set 1: Path Planning Characteristics . . . . .	4-10
4.4.1 Computational Complexity . . . . .	4-10
4.4.2 Unstable Paths . . . . .	4-11
4.5 Test Set 2: MEP-based Control Characteristics . . . . .	4-15
4.5.1 Target Location Relative to Obstacles . . . . .	4-15
4.5.2 Narrow Alley Scenario . . . . .	4-16
4.5.3 Target Designation Change Scenario . . . . .	4-18
4.6 Summary . . . . .	4-20

	Page
V. Conclusions and Recommendations . . . . .	5-1
5.1 Research Goal . . . . .	5-1
5.2 Conclusions . . . . .	5-1
5.3 Recommendations . . . . .	5-3
5.3.1 CLC Optimization in 3 Dimensions . . . . .	5-3
5.3.2 MPC Implementation . . . . .	5-3
5.3.3 Enhanced Waypoint Generation Technique . . . . .	5-4
5.3.4 CLC Assistance to Waypoint Generator . . . . .	5-4
5.3.5 Remove Altitude Bias from Energy Computation . .	5-4
5.3.6 Situational Logic for Target Designation Change Scenario . . . . .	5-5
5.3.7 Streamline Simulation Code for Efficiency . . . . .	5-5
Bibliography . . . . .	BIB-1



## *List of Figures*

Figure		Page
1.1.	Notional scenario with contour map of a mountain - 1 target in view, 1 obstacle in LOS . . . . .	1-2
1.2.	Notional endgame with contour map of a mountain - 1 new target found, target designation updated . . . . .	1-3
2.1.	Block diagram of FOGMA model . . . . .	2-5
2.2.	Block diagram of CV model . . . . .	2-7
2.3.	CLC conceptual trajectory for transition between straight-line seg- ments defined by the waypoints $wp_{previous}$ , $wp_{current}$ , and $wp_{next}$ .	2-10
2.4.	CLC trajectories with $\kappa$ parameter . . . . .	2-14
2.5.	System model of an F-16 aircraft and autopilot controllers . . . . .	2-22
2.6.	Description of flight path and heading angles . . . . .	2-24
3.1.	Flow diagram of simulation . . . . .	3-2
3.2.	Notional waypoint placement - paths do not impact obstacle . . . .	3-5
3.3.	Waypoint extension . . . . .	3-9
3.4.	Extension of notional waypoints . . . . .	3-10
3.5.	Modified simulation model . . . . .	3-24
4.1.	Scenario in which trajectory was forced up-and-over the obstacle - red path is MEP/MDP and blue paths are non-selected candidates . . .	4-9
4.2.	Scenario where the selection of the around path is forced - red path is MEP/MDP, magenta paths are unstable/infeasible, and blue path is an unselected candidate . . . . .	4-10
4.3.	Increasing path complexity - red path is MEP and black path is MDP	4-12
4.4.	Trend data for growth of waypoint generation and path selection times	4-13
4.5.	Scenario where MDP produces unstable system . . . . .	4-14
4.6.	Unstable system response using MDP . . . . .	4-14

Figure		Page
4.7.	Zoomed in view of unstable system response using MDP . . . . .	4-15
4.8.	Target in the open . . . . .	4-16
4.9.	Target located close to obstacle . . . . .	4-17
4.10.	Narrow alley scenario . . . . .	4-17
4.11.	Zoomed in view of narrow alley scenario . . . . .	4-18
4.12.	Target designation change scenario . . . . .	4-19

# *List of Tables*

Table		Page
4.1.	Starting location and target position for basic demonstration . . . .	4-8
4.2.	Path selection cost values for scenario in which trajectory is forced up-and-over the obstacle - Path 1 is up-and-over, Paths 2 and 3 are around, Paths 4 and 5 are up-and-around . . . . .	4-9
4.3.	Path selection cost values for scenario in which trajectory is forced around the obstacle - Path 1 is up-and-over, Paths 2 and 3 are around, Paths 4 and 5 are up-and-around . . . . .	4-11
4.4.	Path complexity data . . . . .	4-11
4.5.	Starting location for munition and target positions for target desig- nation change scenario . . . . .	4-19

# *List of Abbreviations*

Abbreviation	Page
LOS line-of-sight . . . . .	1-1
MPC model predictive control . . . . .	1-2
CLC circle-line-circle . . . . .	1-4
PDF probability density function . . . . .	2-3
FOGMA first-order Gauss-Markov acceleration . . . . .	2-5
CV constant velocity . . . . .	2-5
RHC receding horizon control . . . . .	2-13
MEP Minimum Effort Path . . . . .	4-1
MDP Minimum Distance Path . . . . .	4-1

# A MINIMUM EFFORT CONTROL APPROACH TO GUIDED MUNITION PATH PLANNING

## *I. Introduction*

Over the past decade, smart munitions have played an increasing role in modern conventional warfare. Due to the impressive success of these guided munitions, recent efforts have focused on boosting the agility and efficiency of this class of weapons. One way this performance boost can be achieved is by allowing autonomous in-flight retargeting of the munition such that its flight terminates at the most valuable target in the scenario. This increased capability does not, however, come without its own cost. The munition must now be able to ensure it avoids obstacles that appear in the path to the new target. In addition, once an obstacle-free path is determined, a method must be realized to generate an optimal set of control inputs to navigate along this path.

Despite this desire for greater autonomy and capability, current guidance algorithms typically lack the ability to modify target selection after launch or to avoid obstacles identified by the vehicle's onboard sensors. Standard guidance algorithms simply force the vehicle to fly along the unobstructed line-of-sight (LOS) trajectory between the vehicle and its target while maintaining the predetermined target assignment. In a multiple-target scenario, the terminal waypoint may shift dramatically when the target tracker determines that a new target has the highest value. Any attempt at achieving online retargeting creates a mandate for the guidance algorithm to perform an obstacle avoidance function whenever the terminal waypoint moves. Furthermore, given that mobile friendly and neutral objects may be classified as obstacles, this avoidance function will also be invoked as these objects move about the battlefield. Thus, the need to formulate a guidance algorithm capable of driving the vehicle to the most valuable target still exists.

The basic scenario is shown in Figure 1.1. Initially, since the line-of-sight path between the munition and the target is blocked by a mountain, a path is planned to guide the munition around the obstacle and impact the target. This planned guidance path is

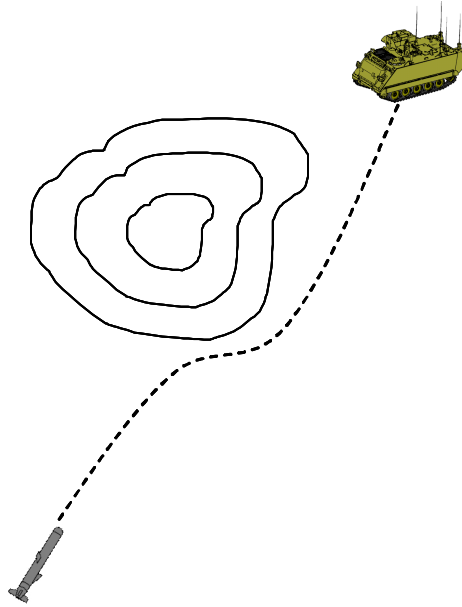


Figure 1.1 Notional scenario with contour map of a mountain - 1 target in view, 1 obstacle in LOS

illustrated by the dotted line between the munition and the troop transport. However, once the munition flies around the side of the mountain and a new target is detected, the entire scene must be reevaluated from the viewpoint of the munition. As a result of this evaluation, one of the potential targets is declared to have the highest value and the munition must determine a new path in order to engage the newly designated target. A depiction of this endgame scenario is shown in Figure 1.2, where the solid red line indicates the path that has already been flown and the munition has determined that the value of the tank is greater than that of the troop transport.

Some researchers have explored nonlinear model predictive control (MPC) as the basis for a method to attain this obstacle avoidance capability [9, 15, 16]. The crux of this approach lies in placing limits on the extent to which the vehicle state (typically position, velocity, acceleration, and other navigation-related states) and control input are allowed to deviate from their nominal values. A constrained optimization problem is then solved in order to generate an input that will fly the vehicle along the optimal path between a set of known waypoints. In general, efforts involving this approach have been limited to a 2-dimensional world. When a third dimension is even considered, the problem

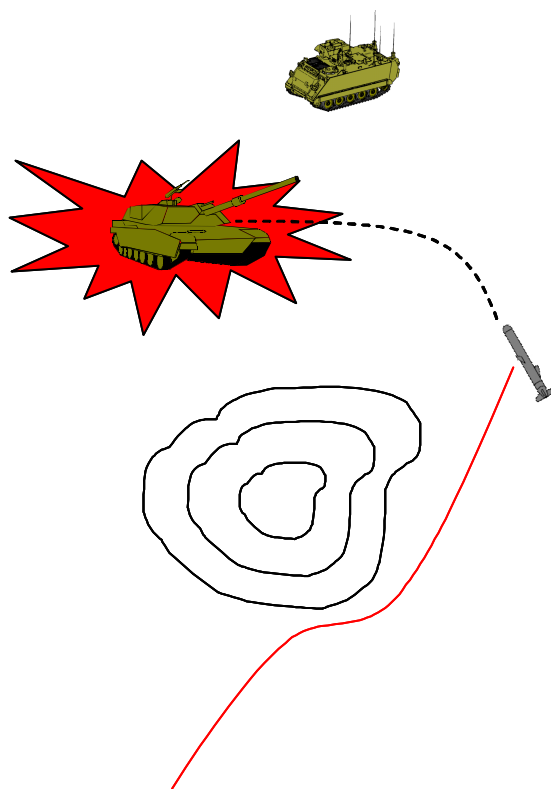


Figure 1.2 Notional endgame with contour map of a mountain - 1 new target found, target designation updated

is reduced back to a 2-dimensional setting in which it is assumed that the vehicle will simply travel either over or around an obstacle, thereby eliminating the opportunity for cooperative motion between the horizontal and vertical planes of the vehicle. For example, if an air vehicle attempts to fly over an obstacle, this MPC approach will maintain the current heading while commanding changes to altitude. Although the desire to extend this approach to a full 3-dimensional world is apparent, the specific manner to accomplish this evolution is not as obvious.

Other researchers have made efforts to utilize a circle-line-circle (CLC) approach to navigating a set of waypoints [1–3, 5, 8, 20]. Essentially, this method constrains the vehicle to fly along the LOS between successive waypoints and attempts to compute the time to begin the turn to the next straight-line segment based on the assumption that the air vehicle will execute all heading changes at a constant turn rate. The flexibility of this approach is realized by its ability to determine a minimum-time path, a path which passes directly through the given waypoint, or any path in between these two which satisfies the assumed system constraints. Current efforts in this arena have only considered 2-dimensional problems (typically the horizontal plane). The extension of this approach to a full 3-dimensional environment presents some interesting issues, such as the need for additional system constraints.

A third, conceptually simple, method employs the principles of minimum effort control to achieve the obstacle avoidance function. When presented with a set of potential waypoints, this control optimization technique can readily determine the path which requires the least amount of applied control energy.

All three of these control optimization methods require a set of desired waypoints as an input to the procedure. Bearing this requisite input in mind, there remains a need for a manner to determine these waypoints, especially when the vehicle operates in a changing environment.

### *1.1 Research Goals and Contributions*

Motivated by the desire to fly an air vehicle toward the highest value target within a scene, this research attempts to generate a 3-dimensional guidance algorithm capable of



navigating the vehicle through a set of waypoints such that obstacles are avoided in an environment in which the designated target is prone to change. Specifically, a multiple-target tracker will be created to provide the munition with information regarding the position, velocity, and value of the various targets within the scenario. Additionally, a path planning method will be developed to formulate a flight path that facilitates obstacle avoidance. These two pieces (multi-target tracker and path planner) comprise the entire guidance algorithm. Finally, the three control optimization techniques (model predictive control, circle-line-circle, and minimum effort control) will each be utilized as the basis of a method to determine the optimal set of control inputs required to have the air vehicle traverse this reference trajectory.

The primary contribution of this research is the utilization of the minimum effort control optimization technique as the basis for a path planning algorithm. Although minimum effort control has been utilized for generating optimal control inputs, the notion of using this control optimization approach as the foundation for a path planner has not been attempted by other researchers in the navigation field. Secondary contributions of this research are the attempts to extend CLC and MPC control optimization techniques to 3-dimensional situations.

## *1.2 Scope and Assumptions*

First and foremost, although the ultimate intention is to utilize a procedure of this nature in online operation, the real-time operation constraint will not be imposed upon this guidance algorithm. This relaxed approach is taken since this research still falls within the realm of concept exploration. Given that this research is only investigating the concept's general feasibility and is not, at present, designated for application on any specific airframe, the tests will be conducted using a six-degree-of-freedom F-16 flight dynamics model (already available in-house) in place of any specific munition model. Additionally, it will be assumed that autopilots capable of maintaining commanded velocity, heading and altitude settings are available for use with the specified air vehicle model. Furthermore, in an effort to reduce the complexity of the problem, it will be assumed that we desire to maintain a constant velocity during each of the simulations. Finally, for ease

of implementation in this early stage of concept exploration, it will be assumed that the munition has knowledge of all the obstacles within the scene, including those that would normally be occluded from view.

### *1.3 Thesis Organization*

Chapter 2 presents the theoretical background information required to understand the development and application of this guidance algorithm. Section 2.2 provides details on target tracking concepts, to include stochastic estimation and linear Kalman filtering (Section 2.2.1) and types of target models (Section 2.2.2). Section 2.3 discusses various methods of control input generation. Section 2.3.1 presents the concept of circle-line-circle control, while Section 2.3.2 details the theory of the model predictive control method. The basic properties of controllability matrices and Grammians are presented in Section 2.3.3 and their applicability to the theory behind minimum effort control is shown in Section 2.3.4. Section 2.4 presents the basic simulation model used throughout this research.

Chapter 3 describes the development of the guidance algorithm and provides details regarding the simulation. Section 3.2 details how the standard multi-target tracker is modified to provide the unique information required by this guidance algorithm. Section 3.3 then discusses the method implemented to generate waypoint sets for the purpose of obstacle avoidance. Section 3.4 describes the application of the minimum effort control technique in solving the problem of selecting the optimal path from the candidate waypoint sets. These three sections provide a concise description of the development of the entire guidance algorithm. Section 3.5 presents the extension of the MPC and CLC control schemes to three dimensions and then demonstrates how minimum effort control is used to generate the commands, to be sent to the vehicle's three autopilots, based on the waypoint information contained within the selected path. Finally, Section 3.6 details the adaptation of the simulation model to accommodate the various innovations of this research.

Chapter 4 presents an analysis of simulation results. The guidance path selected by the path planning algorithm developed in this thesis will be compared with the minimum distance trajectory for that particular test case in order to illustrate the key benefits and shortcomings of each trajectory generation method.

Chapter 5 concludes this research by summarizing the algorithm development of Chapter 3 and the results obtained in Chapter 4. Furthermore, it presents recommendations for future research regarding this concept.

## II. Background

### 2.1 Introduction

The desire to guide an air vehicle toward a moving target immediately invokes the need to have a method to determine the target's state information (position, velocity, ID, etc.). Section 2.2 is devoted to exploring the basics of target tracking. Given that most modern target trackers rely on Kalman filtering to provide estimates of the target's state [6], the details of stochastic state estimation in relation to Kalman filters, and the details of several classes of target models (to be used in the Kalman filter) are presented in this section. Section 2.3 addresses the fundamental concepts of the three control generation methods that will be investigated by this research. A subsection on controllability matrices and Grammians is included to assist the discussion of minimum effort control. Section 2.4 presents the basic F-16 system model which provides the environment for conducting simulations.

### 2.2 Target Tracking Basics

As mentioned in the introduction, the Kalman filter is perhaps the most widely used estimation technique in current target tracking algorithms. Though they are not generally thought of in this fashion, for the purposes of target tracker design, Kalman filters may be split into two pieces. The first part consists of the development of the equations which define the *linear* Kalman filter. This piece is well grounded in the theory of stochastic estimation and the form of the equations generally remains the same from filter to filter. The other portion of the Kalman filter is the user-defined component and primarily entails the selection of a model to describe the motion of the expected target type. Bearing this all in mind, we now turn to the discussion of these two aspects of target tracking.

*2.2.1 Stochastic Estimation and Linear Kalman Filtering.* The development of the linear Kalman filter equations in this section is an adaptation of the material presented in the Maybeck text [12]. This material is intended to be a summary of the key points of linear Kalman filtering and its stochastic estimation underpinnings. The interested reader is directed to [12] for a more in-depth derivation of the filter equations.

Let us assume that we wish to estimate the state values for a linear system which can be described by the following discrete-time state-space model (this may also be an equivalent discrete-time model that was derived from an underlying continuous-time model description, as shown in Section 2.4 of [12]):

$$\begin{aligned}\underline{x}(t_i) &= \Phi(t_i, t_{i-1})\underline{x}(t_{i-1}) + G_d(t_{i-1})\underline{w}_d(t_{i-1}) \\ \underline{z}(t_i) &= H(t_i)\underline{x}(t_i) + \underline{v}(t_i)\end{aligned}\tag{2.1}$$

where  $\underline{x}(t_i)$  is the system state vector and  $\underline{z}(t_i)$  is a noise-corrupted measurement vector of the (potentially time-varying) system state at time sample  $t_i$ .  $\Phi(t_i, t_{i-1})$  is the state transition matrix which describes the homogeneous motion of the system state from time sample  $t_{i-1}$  to  $t_i$ ,  $G_d(t_{i-1})$  is a noise weighting matrix based on the system model description, and  $H(t_i)$  is the measurement matrix which defines the specific combinations of states that are represented by the measurements. The  $\underline{w}_d(t_{i-1})$  and  $\underline{v}(t_i)$  terms represent mutually independent discrete-time white Gaussian noise vectors with the following mean and variance statistics:

$$\begin{aligned}E\{\underline{w}_d(t_i)\} &= E\{\underline{v}(t_i)\} = \underline{0} \\ E\{\underline{w}_d(t_i)\underline{w}_d(t_j)^T\} &= Q_d(t_i)\delta_{ij} \\ E\{\underline{v}(t_i)\underline{v}(t_j)^T\} &= R(t_i)\delta_{ij}\end{aligned}\tag{2.2}$$

where  $Q_d(t_i)$  is the process noise covariance matrix,  $R(t_i)$  is the measurement noise covariance matrix, and  $\delta_{ij}$  is the well known Kronecker delta function. In these expressions, the subscript  $d$  indicates that this is, in fact, either a discrete-time model or the discrete-time equivalent of a continuous-time model. The process and measurement noises are also assumed to be independent of the initial system state vector,  $\underline{x}(t_0)$ . In looking forward to the goal of designing a target tracking filter,  $Q_d(t_i)$  can be thought of as a measure of our confidence in the assumed target dynamics model while  $R(t_i)$  represents the quality of our measuring device (sensor).

Furthermore, let us also assume that the initial state,  $\underline{x}(t_0)$ , is well described by a Gaussian probability density function (PDF) with mean  $\hat{\underline{x}}(t_0)$  and covariance matrix  $P(t_0)$ . It can then be shown that the conditional density function  $f_{\underline{x}(t_i)|Z(t_i)}(\underline{\xi}|Z_i)$ , for the state  $\underline{x}(t_i)$  conditioned on the measurement history up until that time ( $Z(t_i)$ ), remains Gaussian with conditional mean  $\hat{\underline{x}}(t_i^+)$  and conditional covariance  $P(t_i^+)$ . One important distinction between  $Z(t_i)$  and  $\underline{z}(t_i)$  must be made here;  $Z(t_i)$  is the history of all measurements from time  $t_0$  to time  $t_i$ , while  $\underline{z}(t_i)$  is merely the measurement at time  $t_i$ . It can also be proven that the conditional density function  $f_{\underline{x}(t_i)|Z(t_{i-1})}(\underline{\xi}|Z_{i-1})$ , for  $\underline{x}(t_i)$  conditioned on the previous measurement history ( $Z_{i-1}$ ), maintains its Gaussian structure with conditional mean  $\hat{\underline{x}}(t_i^-)$  and conditional covariance  $P(t_i^-)$  [12]. The conditional density function for the propagated state is recognized as  $f_{\underline{x}(t_i)|Z(t_{i-1})}(\underline{\xi}|Z_{i-1})$ , while  $f_{\underline{x}(t_i)|Z(t_i)}(\underline{\xi}|Z_i)$  is the conditional density function for the updated state. This distinction is further emphasized by the superscript notation in which "-" indicates first and second order statistics *before* a measurement update, whereas "+" indicates these same statistics *after* the incorporation of the current measurement. The conditional mean and covariance for both the propagated and the updated states are defined as:

$$\begin{aligned}
\hat{\underline{x}}(t_i^-) &\triangleq E\{\underline{x}(t_i)|Z(t_{i-1}) = Z_{i-1}\} \\
P(t_i^-) &\triangleq E\{[\underline{x}(t_i) - \hat{\underline{x}}(t_i^-)][\underline{x}(t_i) - \hat{\underline{x}}(t_i^-)]^T | Z(t_{i-1}) = Z_{i-1}\} \\
\hat{\underline{x}}(t_i^+) &\triangleq E\{\underline{x}(t_i)|Z(t_i) = Z_i\} \\
P(t_i^+) &\triangleq E\{[\underline{x}(t_i) - \hat{\underline{x}}(t_i^+)] [\underline{x}(t_i) - \hat{\underline{x}}(t_i^+)]^T | Z(t_i) = Z_i\}
\end{aligned} \tag{2.3}$$

When we talk about the Kalman filter's estimate of the target state vector, we will typically be referring to the conditional mean of the updated state. Given that the expected value is a linear operation and that the state transition matrix is a purely deterministic quantity, we are now ready to present the explicit expressions for the conditional mean and covariance.

For the propagated state estimate, we may substitute the assumed form of  $\underline{x}(t_i)$  from Equation (2.1) into the definitions of  $\hat{\underline{x}}(t_i^-)$  and  $P(t_i^-)$  in Equation (2.3) and solve to obtain

$$\begin{aligned}\hat{\underline{x}}(t_i^-) &= \Phi(t_i, t_{i-1})\hat{\underline{x}}(t_{i-1}^+) \\ P(t_i^-) &= \Phi(t_i, t_{i-1})P(t_{i-1}^+)\Phi^T(t_i, t_{i-1}) + G_d(t_{i-1})Q_d(t_{i-1})G_d^T(t_{i-1})\end{aligned}\quad (2.4)$$

In the case of the updated state estimate, we first note that  $\hat{\underline{x}}(t_i^+)$  and  $P(t_i^+)$  are the mean and covariance associated with the conditional PDF  $f_{\underline{x}(t_i)|Z(t_i)}(\underline{x}|Z_i)$ , which may be rewritten as

$$f_{\underline{x}(t_i)|Z(t_i)}(\underline{x}|Z_i) = \frac{f_{\underline{z}(t_i)|\underline{x}(t_i), Z(t_{i-1})}(\underline{z}_i|\underline{x}, Z_{i-1})f_{\underline{x}(t_i)|Z(t_{i-1})}(\underline{x}|Z_{i-1})}{f_{\underline{z}(t_i)|Z(t_{i-1})}(\underline{z}_i|Z_{i-1})}\quad (2.5)$$

after a few applications of Bayes' rule. It can then be shown that, for the case of a *linear* measurement model, this conditional PDF remains Gaussian with mean and covariance given by [12]

$$\begin{aligned}\hat{\underline{x}}(t_i^+) &= \hat{\underline{x}}(t_i^-) + K(t_i)[\underline{z}_i - H(t_i)\hat{\underline{x}}(t_i^-)] \\ P(t_i^+) &= P(t_i^-) - K(t_i)H(t_i)P(t_i^-) \\ K(t_i) &= P(t_i^-)H^T(t_i)[H(t_i)P(t_i^-)H^T(t_i) + R(t_i)]^{-1}\end{aligned}\quad (2.6)$$

where  $K(t_i)$  is referred to as the Kalman gain. At this point, it should be noted that, given the structure of the equations, the entire time history of the Kalman gain and both the propagated and updated covariances may be computed from the system definition and the initial covariance  $P(t_0)$  without requiring access to the actual real-time measurements.

*2.2.2 Target Models.* Now that we have the structure of the linear Kalman filter, we are ready to investigate the other essential part of this target tracking filter: the target dynamics model. A plethora of dynamics models exist, all based on different assumptions about the behavior of the target system. While it might be interesting to investigate the full realm of target models, our intent is only to present the details of a few of the more commonly used model types. To that end, we will present the state transition and

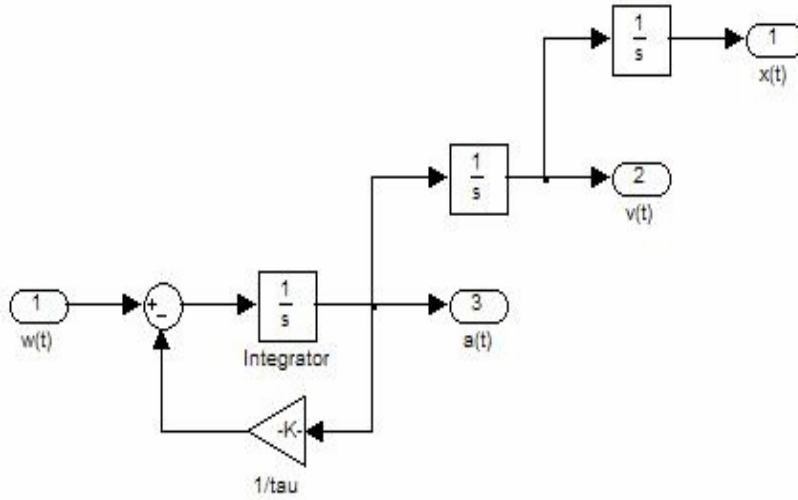


Figure 2.1 Block diagram of FOGMA model

process noise matrices for the first-order Gauss-Markov acceleration (FOGMA) model, the constant velocity (CV) model, and the coordinated turn model.

*2.2.2.1 First-Order Gauss-Markov Acceleration Model.* As the name implies, the FOGMA model assumes that the acceleration of the target is adequately described by a first-order lag system driven by white Gaussian noise (of appropriate strength). In general, this model also assumes that the dimensions of the system (x, y, and z directions for instance) may be decoupled, such that if one were to develop the state transition ( $\Phi$ ) and process noise ( $Q_d$ ) matrices for position, velocity, and acceleration in one direction, then the full dimension system would be described by the use of independent copies of the  $\Phi$  and  $Q_d$  matrices.

This model is shown, conceptually, in Figure 2.1 for the continuous-time case. Based on this model, the continuous-time state equations have the form

$$\begin{aligned}\dot{x}(t) &= v_x(t) \\ \dot{v}_x(t) &= a_x(t) \\ \dot{a}_x(t) &= -\frac{1}{\tau_m}a_x(t) + w_x(t)\end{aligned}\tag{2.7}$$



where  $\tau_m$  is the time constant of the target's maneuver and  $w_x(t)$  is continuous-time zero-mean white Gaussian noise of variance  $\sigma_a^2$  (this is also the mean-squared acceleration). Converting this to a discrete-time representation, we obtain the following state transition matrix [6, 10]

$$\Phi = \begin{bmatrix} 1 & T & \tau_m^2(-1 + \frac{T}{\tau_m} + e^{-T/\tau_m}) \\ 0 & 1 & \tau_m(1 - e^{-T/\tau_m}) \\ 0 & 0 & e^{-T/\tau_m} \end{bmatrix} \quad (2.8)$$

where  $T$  is the sample interval for the discrete-time system and the exact form of the process noise matrix may be found in Section 4.2.1 of [6]. Assuming that the sample interval ( $T$ ) is significantly less than the maneuver time constant ( $\tau_m$ ), we obtain the simplified state transition and process noise matrices [4, 6, 10]

$$\begin{aligned} \Phi &= \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \\ Q_{d,FOGMA} &= \frac{2\sigma_a^2}{\tau_m} \begin{bmatrix} \frac{T^5}{20} & \frac{T^4}{8} & \frac{T^3}{6} \\ \frac{T^4}{8} & \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^3}{6} & \frac{T^2}{2} & T \end{bmatrix} \end{aligned} \quad (2.9)$$

Note that this limiting case of the FOGMA model is actually the definition of the constant acceleration model, in which jerk (the time derivative of acceleration) is modeled as zero-mean white Gaussian noise.

The assumption that  $T \ll \tau_m$  is not a requisite characteristic of the FOGMA model; however, it permits the use of significantly simplified definitions for the  $\Phi$  and  $Q_d$  matrices. In addition, as  $T$  approaches  $\tau_m$ , the accuracy of our acceleration estimate continually degrades until, for  $T \gg \tau_m$ , we essentially lose the ability to track the target's acceleration. At this point, we would be better off switching to a constant velocity model since the acceleration appears white instead of time-correlated.

*2.2.2.2 Constant Velocity Model.* The CV model is based on the assumption that the target's acceleration is adequately described by zero-mean, white Gaussian noise

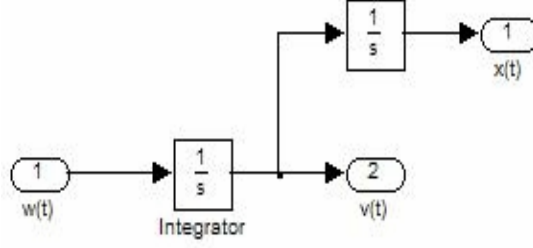


Figure 2.2 Block diagram of CV model

and the velocity essentially remains constant. As shown in Figure 2.2, the model for the velocity state is depicted as an integrator driven by zero-mean white Gaussian noise. Similar to the FOGMA model, the CV model also assumes a decoupling of the target motion in the x, y, and z directions.

Based on the model in Figure 2.2, the continuous-time state equations are

$$\begin{aligned}\dot{x}(t) &= v_x(t) \\ \dot{v}_x(t) &= 0 + w_x(t)\end{aligned}\tag{2.10}$$

The state transition and process noise matrices are then [4,6]

$$\begin{aligned}\Phi &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \\ Q_{d,CV} &= q \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^2}{2} & T \end{bmatrix}\end{aligned}\tag{2.11}$$

where  $q$  is the strength of the zero-mean white Gaussian noise used to define the target's acceleration. As implied in our discussion of the FOGMA model, the CV model is seen to be a limiting case of the FOGMA model for the situation where  $\tau_m$  is significantly smaller than  $T$ .

*2.2.2.3 Coordinated Turn Models.* The two target models discussed so far have assumed that it is possible to decouple a 3-dimensional problem (x, y, z directions)

into three completely separate 1-dimensional problems. This assumption is not always valid, as in the case of an aircraft performing a coordinated turn [6]. In this situation, target motion in one direction will provide information regarding complementary motion in one or both of the remaining directions. This gives rise to the coordinated turn model [6], wherein it is assumed that the turn is accomplished at a nearly constant rate of turn ( $\omega$ ).

For a coordinated horizontal (x,y) turn, where we assume the third direction of motion acts independent of the other two directions, the state transition matrix (corresponding to the state vector  $[x \ v_x \ y \ v_y \ \omega]^T$ ) and the process noise matrix for the coupled directions are given by

$$\Phi = \begin{bmatrix} 1 & \frac{\sin(\omega T)}{\omega} & 0 & \frac{\cos(\omega T)-1}{\omega} & \Phi_{15} \\ 0 & \cos(\omega T) & 0 & -\sin(\omega T) & \Phi_{25} \\ 0 & \frac{1-\cos(\omega T)}{\omega} & 1 & \frac{\sin(\omega T)}{\omega} & \Phi_{35} \\ 0 & \sin(\omega T) & 0 & \cos(\omega T) & \Phi_{45} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{\underline{x}=\hat{\underline{x}}}$$

$$Q_{d,turn} = \sigma_a^2 \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} & 0 & 0 & 0 \\ \frac{T^3}{2} & T^2 & 0 & 0 & 0 \\ 0 & 0 & \frac{T^4}{4} & \frac{T^3}{2} & 0 \\ 0 & 0 & \frac{T^3}{2} & T^2 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sigma_\omega^2 T^2}{\sigma_a^2} \end{bmatrix} \quad (2.12)$$

where the elements of  $\Phi$  are evaluated at the current state vector estimate,  $\sigma_a^2$  is the variance of the random acceleration, and  $\sigma_\omega^2$  is the turn rate variance. The elements in the final column of the state transition matrix are the partial derivatives of the non-linear discrete-time function with respect to the turn rate and are given by

$$\begin{aligned} \Phi_{15} &= \frac{(\omega T \cos(\omega T) - \sin(\omega T)) v_x - (\omega T \sin(\omega T) - 1 + \cos(\omega T)) v_y}{\omega^2} \\ \Phi_{25} &= -(T \sin(\omega T) v_x + T \cos(\omega T) v_y) \\ \Phi_{35} &= \frac{(\omega T \sin(\omega T) - 1 + \cos(\omega T)) v_x + (\omega T \cos(\omega T) - \sin(\omega T)) v_y}{\omega^2} \\ \Phi_{45} &= T \cos(\omega T) v_x - T \sin(\omega T) v_y \end{aligned} \quad (2.13)$$

## 2.3 Control Concepts

In this section, we delve into a few different control optimization techniques. Given that one of the goals of this research is to evaluate the utility of the CLC and MPC approaches in a 3-dimensional setting, we will first present the background of the work that has been accomplished using these control methods for 2-dimensional problems. Our brief discussion of control concepts will conclude with a presentation on the theory of minimum effort control.

*2.3.1 Circle-Line-Circle Control.* The driving concern behind the CLC control optimization method [1,2] is that there are limits imposed on the magnitude of the control inputs and there is a desire to minimize deviations from a LOS path between waypoints. When a non-zero input is applied, the desire is to have the magnitude of the optimal input be equal to an upper bound. By employing this proposed control scheme, a vehicle would traverse a set of waypoints by following the straight-line path between successive waypoints until a certain time (known as the switching time), when a series of turns are executed at the vehicle's maximum turn rate, in order to reach the next straight-line segment. A notional view of the trajectory generated by this manner of control is shown in Figure 2.3. Note that in order to execute a sharp turn to the right at  $\underline{wp}_{current}$ , one would first command a short turn to the left in an effort to smooth the transition into the rightward turn. A similar smoothing turn would be executed at the end of the turn in order to transition back to the straight-line segment. All that remains is a need to determine the switching time, and then one would have an optimal control method that incorporates constraints on the control input. Currently, this method has only been applied to 2-dimensional problems.

Since the theory of this control mechanism relies heavily on Pontryagin's Minimum Principle, we shall start off by outlining the details of this principle [1, 19]. Let us adopt the form of the general nonlinear dynamic system given by

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \underline{u}(t)) \quad (2.14)$$

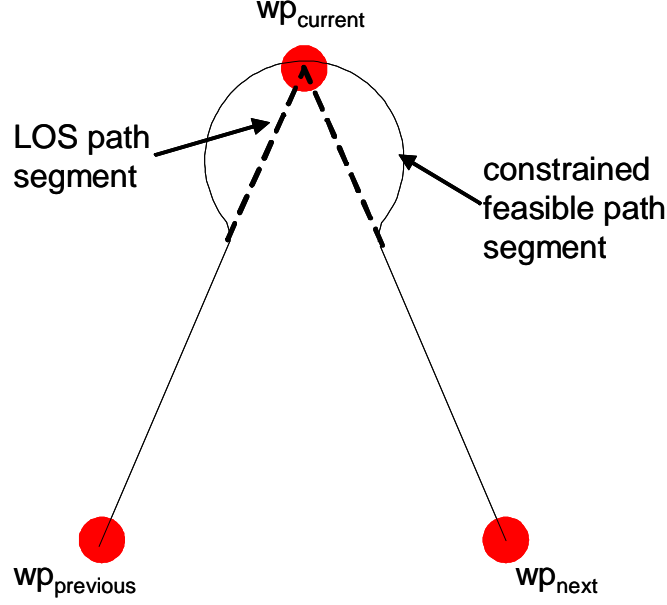


Figure 2.3 CLC conceptual trajectory for transition between straight-line segments defined by the waypoints  $wp_{previous}$ ,  $wp_{current}$ , and  $wp_{next}$

with initial condition  $\underline{x}(t_0)$ , final state constraint set

$$\chi = \{\underline{x}(t_f) \in \mathbb{R}^{n_x} | \underline{\Theta}(\underline{x}) = \underline{0}\} \quad (2.15)$$

and input constraint set

$$\mu = \{\underline{u} \in \mathbb{R}^{n_u} | \underline{\Xi}(\underline{u}) \geq \underline{0}\} \quad (2.16)$$

for the vectors of continuously differentiable functions defined by  $\underline{f}$ ,  $\underline{\Theta}$  and  $\underline{\Xi}$ . For the functions in the constraint sets, let us assume that the gradient matrices,  $\frac{\partial \underline{\Theta}(\underline{x})}{\partial \underline{x}}$  and  $\frac{\partial \underline{\Xi}(\underline{u})}{\partial \underline{u}}$ , consist of linearly independent gradient vectors for all possible  $\underline{x}$  and  $\underline{u}$ , respectively. Now, if  $\underline{u}^*$  is an admissible and optimal control input, optimal in the sense that it minimizes the cost function

$$J = \int_0^{t_f} dt C(\underline{x}(t), \underline{u}(t)) \quad (2.17)$$

then a continuous piecewise-differentiable function  $\underline{\gamma}(t) = [\gamma_1 \cdots \gamma_{n_x}]^T$ , a constant  $\gamma_0 \geq 0$ , and a constant vector  $\underline{\rho} = [\rho_1 \cdots \rho_{n_g}]^T$  are guaranteed to exist and satisfy the following

conditions [1]:

$$\dot{\underline{\gamma}}^T = -\frac{\partial H}{\partial \underline{x}}|_{\underline{u}=\underline{u}^*} \quad (2.18a)$$

$$\underline{\Theta}(\underline{x}(t_f)) = \underline{0} \quad (2.18b)$$

$$\underline{\gamma}^T(t_f) = \underline{\rho}^T \frac{\partial \underline{\Theta}(\underline{x})}{\partial \underline{x}}|_{\underline{x}=\underline{x}(t_f)} \quad (2.18c)$$

$$\gamma_0 \neq 0 \quad (2.18d)$$

$$\underline{\gamma} \neq \underline{0} \quad (2.18e)$$

where  $H$  is the system Hamiltonian given by

$$H(\underline{x}(t), \underline{u}(t), \underline{\gamma}(t), \gamma_0) = \gamma_0 C(\underline{x}(t), \underline{u}(t)) + \underline{\gamma}^T(t) \underline{f}(\underline{x}(t), \underline{u}(t)) \quad (2.19)$$

Furthermore, the Hamiltonian takes on a global minimum almost everywhere when evaluated along the trajectory resulting from the application of the optimal input:

$$H(\underline{x}(t), \underline{u}^*(\underline{x}(t)), \underline{\gamma}(t), \gamma_0) = \min_{\underline{u} \in \mu} (H(\underline{x}(t), \underline{u}(t), \underline{\gamma}(t), \gamma_0)) = 0 \quad (2.20)$$

Note that Equation (2.18a) is a backward time propagation for the adjoint state,  $\underline{\gamma}(t)$ , and Equation (2.18c) is the terminal condition for this propagation. Caution must be exercised when applying the minimum principle in that the conditions in Equation (2.18) are only necessary conditions. No guarantee is made that any input which satisfies these four conditions is in fact an optimal control. However, all inputs which satisfy the conditions of Equation (2.18) and Equation (2.20) are referred to as extremal controls and, collectively, these extremal controls comprise the set of possible optimal controls for a given problem. With the foundation of the minimum principle of Pontryagin, the research in [1] proved that commanding an input that has magnitude which is equal to the upper limit of the control constraint produced an extremal input for the class of problems under consideration.

We will now summarize this development for the 2-dimensional navigation problem. The problem begins with a simple set of kinematic equations for the motion of a vehicle

in a horizontal plane

$$\begin{aligned}\dot{x}(t) &= s \cos(\Psi(t)) \\ \dot{y}(t) &= s \sin(\Psi(t)) \\ \dot{\Psi}(t) &= u(t)\end{aligned}\tag{2.21}$$

where  $x$  and  $y$  are position coordinates,  $s$  is a constant speed and  $\Psi$  is the vehicle heading angle. The magnitude of the input is constrained, such that

$$\|u\| \leq u_{max}\tag{2.22}$$

to account for the fact that the vehicle is not capable of making arbitrarily fast turns. Next, it is desired to navigate a set of known waypoints in a time-optimal fashion while obeying the input constraint. Since we are considering a time minimization as our criterion of optimality, the appropriate cost function to be minimized is simply

$$J = \int_0^{t_f} dt\tag{2.23}$$

From this point, the proof of an extremal control becomes the evaluation of the inequality implied by Equation (2.20)

$$\begin{aligned}H(\underline{x}, u^*, \underline{\gamma}, \gamma_0) &\leq H(\underline{x}, u, \underline{\gamma}, \gamma_0), \quad \forall u \in \mu \\ H(\underline{x}, u, \underline{\gamma}, \gamma_0) &= \gamma_0 + \gamma_1(t)s \cos(\psi) + \gamma_2(t)s \sin(\psi) + \gamma_3(t)u(t)\end{aligned}\tag{2.24}$$

which simplifies to

$$\gamma_3(t)u^*(t) \leq \gamma_3(t)u(t), \quad \forall u \in \mu\tag{2.25}$$

Given this final inequality, the proposed optimal input can be written as

$$u^*(t) = -\text{sign}(\gamma_3(t))u_{max}\tag{2.26}$$

and can be shown to be an extremal control by evaluating the conditions in Equation (2.18).

An additional result of the Anderson research [1] was the development of a convenient method to force the trajectory to pass through a point,  $\underline{p}_{cross}$ , near the current waypoint,  $\underline{wp}_{current}$ . This desired point is constrained to lie on the bisector of the straight-line segments defined by  $\underline{wp}_{previous}$ ,  $\underline{wp}_{current}$ , and  $\underline{wp}_{next}$ . Furthermore, the range of possible locations for  $\underline{p}_{cross}$  is bounded by  $\underline{wp}_{current}$  and a predetermined minimum point,  $\underline{p}_{min}$ . The family of allowable trajectories, as shown in Figure 2.4, is characterized by a  $\kappa$  parameter such that the point where the trajectory crosses the bisector is given by

$$\begin{aligned}\underline{p}_{cross} &= (1 - \kappa)\underline{wp}_{current} + \kappa\underline{p}_{min} \\ \kappa &= \frac{\left\| \underline{wp}_{current} - \underline{p}_{cross} \right\|}{\frac{s}{u_{max}} \left( \frac{1}{\sin(\frac{\beta_{CLC}}{2})} - 1 \right)}\end{aligned}\tag{2.27}$$

where  $\beta_{CLC}$  is the angle between consecutive straight-line segments and  $\kappa \in [0, 1]$ . The trajectories shown in Figure 2.4 represent the set of *possible* paths to fly when executing a constrained turn. In general, we would prefer to apply the constrained turn trajectory obtained by selecting  $\kappa = 1$ , since this option produces a time-optimal solution. The switching time computation for this general set of trajectories may be found in [1].

**2.3.2 Model Predictive Control.** The MPC approach [15] is a quadratic cost minimization technique that falls under the larger heading of receding horizon control (RHC) methods [13]. It allows us to compute an optimal set of controls (valid for a finite length of time) to navigate between a set of known waypoints while explicitly addressing state and input constraints. This optimal set of controls is realized as an optimal perturbation about an assumed nominal input. Receding horizon control methods seek to determine the optimal control,  $\underline{u}^*(t)$ , for the non-linear system of Equation (2.14) by minimizing the following cost function [13] over the finite length interval from time  $t$  to time  $t + T$ :

$$J(\underline{x}, t; \underline{u}) \triangleq \frac{1}{2} \int_t^{t+T} [\underline{x}^T(\tau)\Gamma\underline{x}(\tau) + \underline{u}^T(\tau)\Upsilon\underline{u}(\tau)]d\tau\tag{2.28}$$



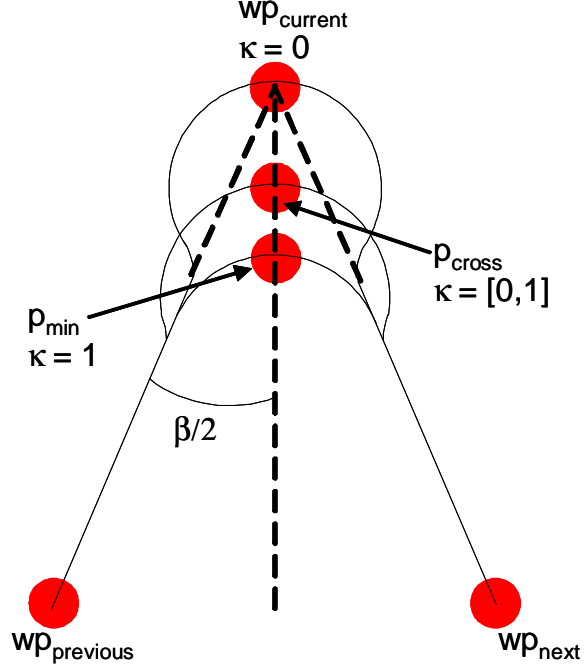


Figure 2.4 CLC trajectories with  $\kappa$  parameter

Within the framework of RHC, model predictive control operates by first obtaining a perturbational linearization of the system about some nominal control input,  $\underline{u}_{nom}(t)$ , and system output,  $\underline{y}_{nom}(t)$ , trajectories such that

$$\begin{aligned}\underline{u}(t) &= \underline{u}_{nom}(t) + \delta\underline{u}(t) \\ \underline{y}(t) &= \underline{y}_{nom}(t) + \delta\underline{y}(t)\end{aligned}\tag{2.29}$$

Next, a set of control basis functions is determined so that the perturbation input and output terms may be rewritten as linear combinations of a single set of scale factors [9, 15]

$$\begin{aligned}\delta\underline{u} &= \underline{U}\underline{\eta} \\ \delta\underline{y} &= \underline{Y}\underline{\eta}\end{aligned}\tag{2.30}$$

where the matrix  $\underline{U}$  describes the control basis functions,  $\underline{\eta}$  is the vector of scale factors, and the matrix  $\underline{Y}$  describes the relationship between the output and the control basis functions. Note that the  $\underline{U}$  and  $\underline{Y}$  matrices are sized such that they contain the required

information for *each* sample instant on the interval from time  $t$  to time  $t+T$ . For example, in the case of a single input described by five basis functions, the dimension of the  $U$  matrix which covers a length- $N$  time interval is  $N \times 5$ . The form of  $U$  is dictated by the choice of control basis functions, and dramatic changes will be observed based on the choice of the basis functions. For instance, the basis functions may be as simple as a set of ramp [15] or tent [9] functions or as complex as a set of Laguerre or Legendre polynomials [16]. The matrix  $Y$  is determined by driving the non-linear system with the input basis functions one at a time.

Once we have obtained the  $U$  and  $Y$  matrices, MPC seeks to minimize the following cost function (shown for a notional discrete-time system)

$$J_k = \sum_{i=k}^{k+N-1} [(\underline{y}_{i+1} - \underline{r}_{i+1})^T \Gamma (\underline{y}_{i+1} - \underline{r}_{i+1}) + \underline{u}_i^T \Upsilon \underline{u}_i] \quad (2.31)$$

where  $N$  is the length of the finite interval horizon,  $\underline{r}_i$  is a reference output trajectory at time sample  $i$ , and  $\Gamma$  and  $\Upsilon$  are cost weighting matrices. Based upon the perturbational linearization of the system, we may rewrite the cost function as

$$\begin{aligned} J_k = \sum_{i=k}^{k+N-1} [ & (\underline{y}_{nom,i+1} - \underline{r}_{i+1})^T \Gamma (\underline{y}_{nom,i+1} - \underline{r}_{i+1}) + \underline{u}_{nom,i}^T \Upsilon \underline{u}_{nom,i} + \delta \underline{y}_{i+1}^T \Gamma \delta \underline{y}_{i+1} \\ & + \delta \underline{u}_i^T \Upsilon \delta \underline{u}_i + 2(\underline{y}_{nom,i+1} - \underline{r}_{i+1})^T \Gamma \delta \underline{y}_{i+1} + 2\underline{u}_{nom,i}^T \Upsilon \delta \underline{u}_i] \end{aligned} \quad (2.32)$$

Noticing that the first two terms represent an incurred cost that is independent of the optimization and remembering the unique definition of the dimensions of the  $U$  and  $Y$  matrices, we may drop the terms that do not involve perturbation variables and simplify the cost function to be

$$J_k = \underline{\eta}^T (Y^T \Gamma Y + U^T \Upsilon U) \underline{\eta} + 2[(\underline{y}_{nom} - \underline{r})^T \Gamma Y + \underline{u}_{nom}^T \Upsilon U] \underline{\eta} \quad (2.33)$$

where the columns of  $\underline{r}$  are the individual time samples,  $\underline{r}_i$ , of the reference trajectory that spans the time interval of interest, and similar definitions apply to  $\underline{y}_{nom}$  and  $\underline{u}_{nom}$ .

Note that the summation in Equation (2.32) is embedded in the multiplications of the higher-dimensioned vectors in Equation (2.33).

In preparation to solve the constrained optimization problem, we must first transform the constraints on the input and output into constraints on the basis function scale factors, since the optimization problem is actually considering the *perturbational* inputs and outputs, as defined by Equation (2.30), instead of the full-scale values. This is a very simple process since the original constraints are given as

$$\begin{aligned} \underline{u}_{lower} &\leq \underline{u} \leq \underline{u}_{upper} \\ \underline{y}_{lower} &\leq \underline{y} \leq \underline{y}_{upper} \end{aligned} \tag{2.34}$$

We may use Equations (2.29) and (2.30) to reform these constraints as upper limits on the basis function scale factors

$$\begin{bmatrix} U \\ -U \\ Y \\ -Y \end{bmatrix} \underline{\eta} \leq \begin{bmatrix} \underline{u}_{upper} - \underline{u}_{nom} \\ \underline{u}_{nom} - \underline{u}_{lower} \\ \underline{y}_{upper} - \underline{y}_{nom} \\ \underline{y}_{nom} - \underline{y}_{lower} \end{bmatrix} \tag{2.35}$$

The MPC problem is now in a format that can be solved by the MATLAB<sup>®</sup> quadratic programming function, *quadprog*.

In general, research on the use of MPC techniques for flight control optimization has been limited to 2-dimensional cases, for which it is assumed that an autopilot is capable of holding the third dimension of motion stable, and a simplified 3-degree-of-freedom model is used to represent the air vehicle. Attempts have been made at considering 3-dimensional scenarios and using a full 6-degree-of-freedom air vehicle model, but these efforts typically revert to 2-dimensional approaches when faced with the task of obstacle avoidance [9].

2.3.3 *Controllability Matrices and Grammians.* We typically consider state space representations of linear (or linearized) systems to be given in the following form

$$\begin{aligned}\dot{\underline{x}}(t) &= A\underline{x}(t) + B\underline{u}(t) \\ \underline{y}(t) &= C\underline{x}(t) + D\underline{u}(t)\end{aligned}\tag{2.36}$$

where  $\underline{x}(t)$  is the length- $n$  state vector,  $\underline{u}(t)$  is the length- $m$  input vector and  $\underline{y}(t)$  is the length- $p$  output vector. The  $A$ ,  $B$ ,  $C$  and  $D$  matrices are the appropriately sized matrices which describe the input-output behavior of the system relative to the internal state vector. In general, the system matrices are functions of time, but for the purpose of this research, we have restricted our attention to constant matrices.

One primary concern for control engineering is to answer the question [14], "Starting from the origin in our state-space, can I drive the state to any desired final location in state-space by applying a control input?" If the answer to this controllability question is yes, then the system is considered to be completely controllable. On the other hand, if the answer is no, then the system is said to be uncontrollable and a secondary concern is created. For an uncontrollable system, we seek to determine the regions of state-space we can in fact reach. Finally, for both the completely controllable and the uncontrollable systems, we must consider the relative cost, in terms of control input energy, required to travel along the various directions of state-space. This last piece brings in a quality aspect to the controllability discussion. We shall now investigate these concerns.

The mathematical equivalent of the controllability question is the desire to know if

$$\underline{x} \in \mathcal{R}\{M_c\}\tag{2.37}$$

is satisfied for all possible  $\underline{x}$ , where  $M_c$  is a special matrix known as the controllability matrix,  $\mathcal{R}\{\cdot\}$  denotes the range space of the bracketed matrix, and  $\underline{x}$  is the state at which we wish to arrive, having started from the zero state. This controllability matrix, of

dimension  $n \times nm$ , is computed from the system matrices in the following manner [7]:

$$M_c = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \quad (2.38)$$

The system is deemed completely controllable if and only if  $M_c$  is of rank  $n$ . In the event that the specified system is uncontrollable, the reachable region of state-space may be obtained by computing the range space of  $M_c$ . Conversely, if one is interested in explicitly specifying the unreachable portion of state-space, then this is given by the null space of  $M_c$ . Methods for performing these two computations may be found in most texts on linear systems or linear algebra [14, 18].

Having addressed the first two issues of controllability, we now consider the energy cost required to drive the system along certain directions of state-space. A convenient construct to perform this investigation already exists in the form of the controllability Grammian. For this presentation, we will need to make use of the state transition matrix, which may be calculated for the case of a time-invariant system via the following equation:

$$\Phi(t - t_0) = e^{A(t-t_0)} \quad (2.39)$$

where  $t_0$  is the initial time. The controllability Grammian ( $W_c$ ) is then defined as [12]

$$W_c(t) \triangleq \int_0^t \Phi(t - \tau) B B^T \Phi^T(t - \tau) d\tau \quad (2.40)$$

or in an alternate, yet equivalent, form as [14]

$$W_c(t) \triangleq \Phi(t) \left( \int_0^t \Phi(-\tau) B B^T \Phi^T(-\tau) d\tau \right) \Phi^T(t) \quad (2.41)$$

and has the following properties [14]

1.  $W_c(t)$  is symmetric for all  $t \geq 0$ .
2. All of the eigenvalues of  $W_c(t)$  are either positive or zero.
3. The system is completely controllable if and only if  $W_c(t)$  is of rank  $n$  for  $t > 0$ .

This is equivalent to requiring  $W_c(t)$  to have only positive eigenvalues.

4.  $W_c(t) = \int_0^t \Phi(\eta) B B^T \Phi^T(\eta) d\eta$  (note that this obtained by substituting  $\eta = t - \tau$ ).
5.  $W_c(t)$  is the solution to the  $n \times n$  matrix Lyapunov differential equation with zero boundary conditions

$$\frac{dW_c(t)}{dt} = A W_c(t) + W_c(t) A^T + B B^T \quad (2.42)$$

6. If the system is stable, then the steady-state controllability Grammian,  $W_{c_{ss}}$ , exists and is the solution to the  $n \times n$  matrix Lyapunov equation

$$0 = A W_{c_{ss}} + W_{c_{ss}} A^T + B B^T \quad (2.43)$$

7. To arrive at the desired state,  $\underline{x}$ , the norm squared value of the minimum energy control satisfies

$$\|\underline{u}^*\|^2 \leq \frac{1}{\lambda_{min}} \|\underline{x}\|^2 \quad (2.44)$$

where  $\lambda_{min}$  is the minimum eigenvalue of  $W_c(t)$ .

8. To arrive at the state given by  $\underline{x} = \alpha_i \underline{\xi}_i$ , where  $\underline{\xi}_i$  is the unit norm eigenvector of  $W_c(t)$  corresponding to the eigenvalue  $\lambda_i$ , the norm squared value of the minimum energy control is given by

$$\|\underline{u}^*\|^2 = \frac{\alpha_i^2}{\lambda_i} \quad (2.45)$$

From the final property in this list, we can see the beginnings of how a concept of quality of controllability may be realized. The best method to describe this is with an example. Let us assume that the eigenvalues for a  $2 \times 2$  steady-state controllability Grammian are  $\lambda_1 = 4$  and  $\lambda_2 = 0.25$ . The energy of the minimum effort control for  $\underline{x} = \underline{\xi}_1$  is  $\|\underline{u}^*\|^2 = 0.25$ , while  $\|\underline{u}^*\|^2 = 4$  for the case of  $\underline{x} = \underline{\xi}_2$ . This demonstrates that movement in the  $\underline{\xi}_2$  direction of state-space is sixteen times as costly as movement in the  $\underline{\xi}_1$  direction. Thus, even though it is possible to move in the  $\underline{\xi}_2$  direction, it is rather difficult to accomplish this movement, relative to the effort required to move in the  $\underline{\xi}_1$  direction.

*2.3.4 Minimum Effort Control.* Given the mathematical construct of the controllability Grammian, we may turn our attention to the concept of minimum effort control. The goal of this control technique is to find the input,  $\underline{u}^*(t)$ , that drives the zero state response

$$\underline{x}_{zs}(t_f) = \Phi(t_f) \int_0^{t_f} \Phi(-\tau) B \underline{u}(\tau) d\tau \quad (2.46)$$

to the desired state  $\underline{x}$  within a finite amount of time, such that the norm of this input is smaller than the norm of any other input that can reach the solution  $\underline{x}_{zs}(t_f) = \underline{x}$ . A control input that attains this final state, one that has been proven to have the minimum norm, is given by [14]

$$\underline{u}^*(t) = B^T \Phi^T(t_f - t) W_c^{-1}(t_f) \underline{x}, \quad 0 < t \leq t_f \quad (2.47)$$

## *2.4 F-16 Simulation Model*

The final topic in this presentation of background information is a discussion of the F-16 dynamics model which provides the basis for the simulation environment. This simulation model was developed by the faculty of the Air Force Institute of Technology (in collaboration with Environmental Tectronics Corporation) [11] and is based on the air vehicle and controller models described in [17].

Figure 2.5 shows the overall setup of the non-linear system, to include the aircraft plant model and the feedback autopilot controllers. At the core of this simulation is a function which applies saturation limits on the inputs and numerically computes the derivative of the state vector. The state and control input vectors for this model are

defined as

$$\underline{x} = \begin{bmatrix} V_T \\ \alpha_{aoa} \\ \beta \\ \phi \\ \theta \\ \psi \\ \phi_{rate} \\ \theta_{rate} \\ \psi_{rate} \\ p_n \\ p_e \\ h \\ engine\ lag \\ ele_{defl} \\ ail_{defl} \\ rdr_{defl} \end{bmatrix} \quad \underline{u} = \begin{bmatrix} throttle \\ ele_{cmd} \\ ail_{cmd} \\ rdr_{cmd} \end{bmatrix} \quad (2.48)$$

where  $V_T$  is the vehicle's air speed,  $\alpha_{aoa}$  is the angle of attack,  $\beta$  is the sideslip angle,  $\phi$ ,  $\theta$ , and  $\psi$  are the roll, pitch, and yaw orientation angles,  $\phi_{rate}$ ,  $\theta_{rate}$ , and  $\psi_{rate}$  are the rates of change of the orientation angles,  $p_n$  and  $p_e$  are the northward and eastward position,  $h$  is the altitude, *engine lag* is a lag state associated with the engine thrust dynamics, and  $ele_{defl}$ ,  $ail_{defl}$ , and  $rdr_{defl}$ , are the actuator deflection angles for the elevator, aileron, and rudder. The angle of attack and the sideslip angle describe the orientation of the vehicle with respect to the velocity vector and are two of the angles which help to define the direction of the vehicle's motion. In the control input vector, *throttle* is commanded throttle setting expressed as a percentage of the maximum value, and  $ele_{cmd}$ ,  $ail_{cmd}$ , and  $rdr_{cmd}$  are the commanded deflection angles for the elevator, aileron, and rudder.

Each of the feedback controllers in Figure 2.5 is a simple autopilot designed to maintain a predetermined set of equilibrium velocity and orientation conditions. External inputs are available to modify the feedback commands in an effort to mimic the inputs



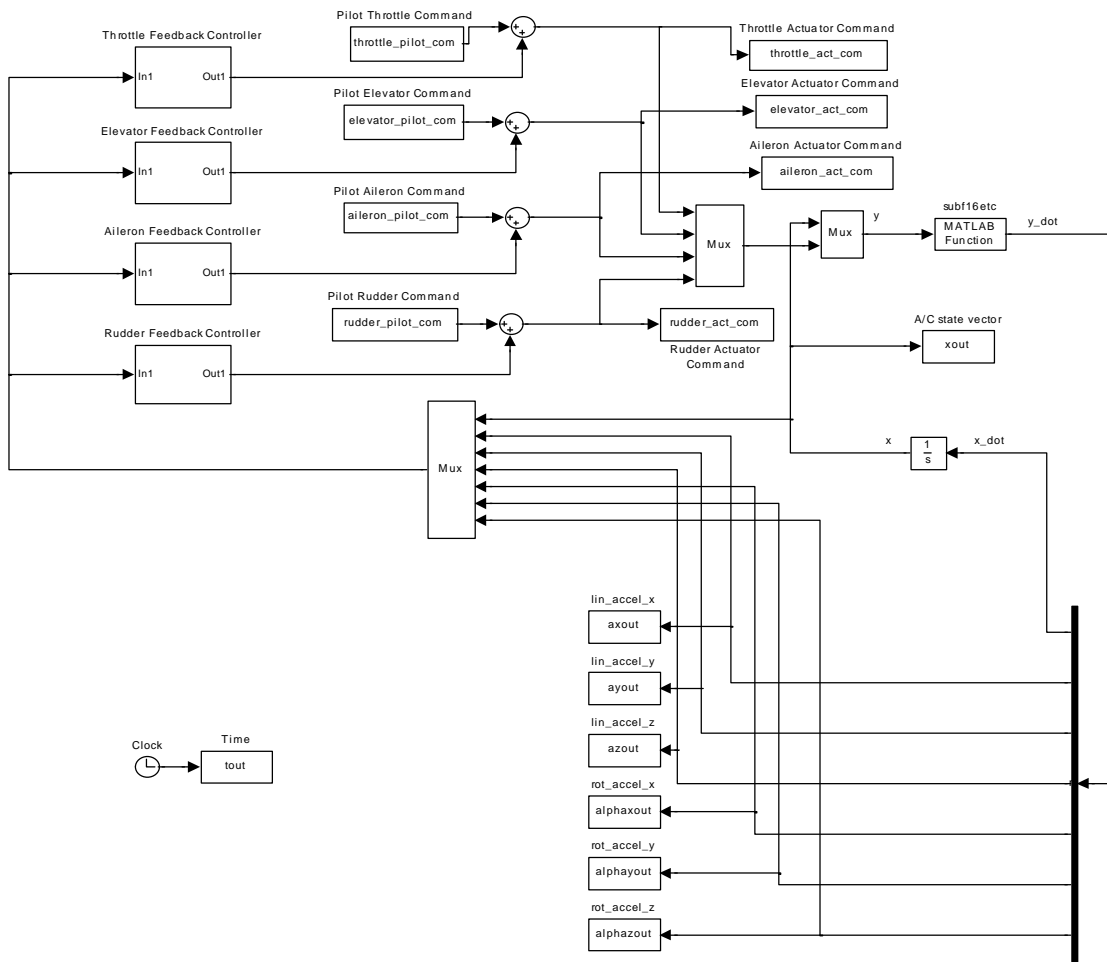


Figure 2.5 System model of an F-16 aircraft and autopilot controllers

of a pilot. This model was originally created to evaluate the forces sensed at the pilot's seat [11] and, as such, the linear (lin\_accel) and rotational (rot\_accel) accelerations shown in Figure 2.5 are holdovers from the research which previously used this model.

In order to facilitate the development in Chapter 3, we must detail the two angles which define the vehicle's velocity vector. These two angles, known as the flight path angle,  $\gamma_{path}$ , and the heading angle,  $\Psi$ , angles are shown in Figure 2.6 and may be calculated using the information from the state vector that defines the system in Figure 2.5. The flight path angle is what actually determines whether the munition is climbing or diving. Knowledge of the angle of attack and the vehicle pitch angle, both of which may be measured, permits us to determine the flight path angle. Mathematically, the flight path angle is computed as

$$\gamma_{path} = \theta - \alpha_{aoa} \quad (2.49)$$

as seen in Figure 2.6. The heading angle describes the motion of the vehicle within the horizontal plane and may be determined from the yaw orientation angle and the sideslip angle using the following expression

$$\Psi = \psi - \beta \quad (2.50)$$

also as seen in Figure 2.6. Note the difference between  $\Psi$ , the heading angle, and  $\psi$ , the vehicle yaw angle. Given these two angles and the air speed of the vehicle we may define the north, east, and vertical velocities as

$$\begin{bmatrix} V_{north} \\ V_{east} \\ V_{vertical} \end{bmatrix} = \begin{bmatrix} V_T \cos(\Psi) \cos(\gamma_{path}) \\ V_T \sin(\Psi) \cos(\gamma_{path}) \\ V_T \sin(\gamma_{path}) \end{bmatrix} \quad (2.51)$$

## 2.5 Summary

The various sections of this chapter have provided the fundamental concepts required to proceed with this research. First, Section 2.2 provided the background of linear Kalman

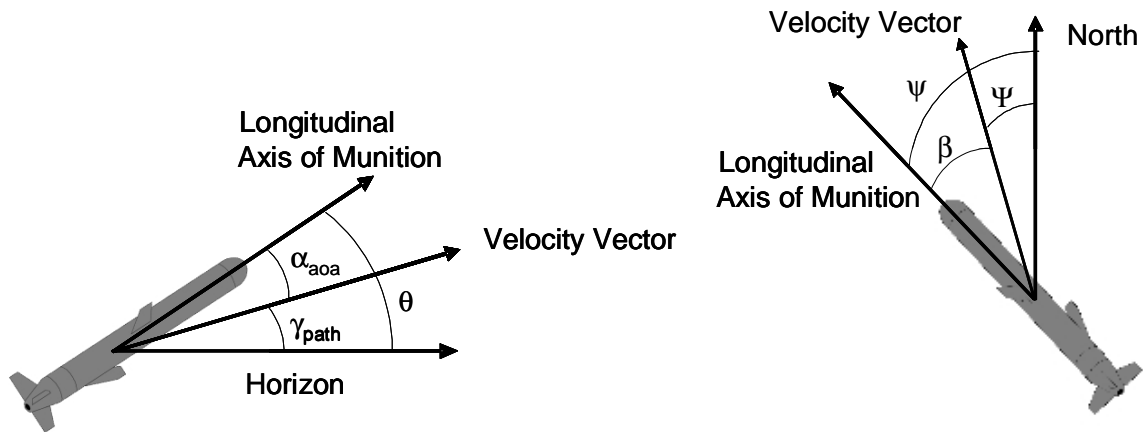


Figure 2.6 Description of flight path and heading angles

filtering, the popular method used in many target tracking applications. Next, Section 2.3 detailed a few methods, including circle-line-circle control, model predictive control, and minimum effort control, to accomplish an optimal control task in two dimensions. Finally, Section 2.4 presented a basic simulation environment based on the flight dynamics of an F-16 aircraft. Equipped with this foundation, we now turn to the development of a 3-dimensional guidance algorithm and the extension of the control optimization techniques discussed in Section 2.3.

### *III. Algorithm Development and Simulation Description*

#### *3.1 Introduction*

In this chapter we present the development of the guidance algorithm and the extension of the control optimization techniques to three dimensions. As shown in Figure 3.1, the guidance algorithm consists of the target tracking, waypoint generation, and path selection subroutines. When appropriate, pseudocode will be provided and an example will be discussed in the text. Additionally, the rationale behind major decisions will be explained. The target information block of Figure 3.1 is merely an external input which provides the target measurement data required by the target tracker. We are now ready to step through the remaining sections of Figure 3.1 sequentially, starting with the target tracker.

#### *3.2 Target Tracker*

The primary function of the target tracker in Figure 3.1 is to determine the target with the highest value and to compute an aimpoint for the munition. In order to accomplish these tasks, the tracker will have to maintain accurate position and velocity estimates for each potential target. Additionally, we will require some form of attribute data so that the tracker can classify the potential targets according to a list of target types.

The linear Kalman filter of Equations (2.4) and (2.6) forms the basis of this target tracker. Realizing that we intend to engage ground targets, we use the constant velocity model, with  $\Phi$  and  $Q_d$  matrices given by Equation (2.11), to describe the kinematics of the targets in the scenario. For simplicity, each target in this multi-target scene will be tracked independently, and only the measurement corresponding to the current target under consideration will be presented to the Kalman filter. The estimate that we obtain ( $\hat{x}$ ) from this tracking filter only contains information regarding the position and velocity of the target. An attribute tracking method is required to provide the determination of target type. A hard decision on target type is not necessary, and it is possible to make soft assignments, such that we obtain the probability of the target being a specific type. As will be shown in the following discussion on target values, the soft assignment of target

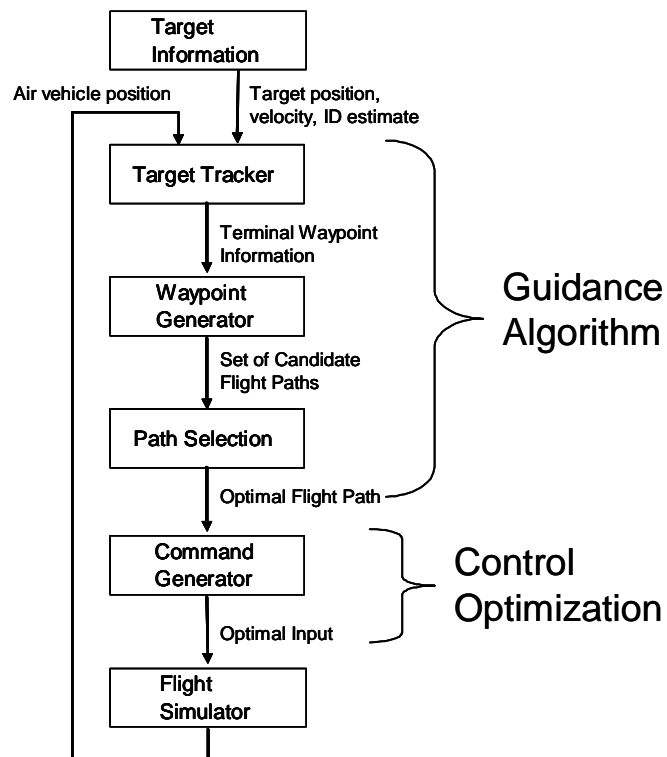


Figure 3.1 Flow diagram of simulation

type is preferred since it permits the computation of a blended target value. In order to ensure we only engage enemy targets, friendly and neutral targets would have to be considered obstacles to be avoided by the remainder of the path planning process.

Given our desire to engage the highest value target, we must now consider how to assign target values. Two obvious possibilities for computing target value are a *maximum a posteriori* value computation

$$v_1^i = \max_{p_{T_j}^i} \nu_{T_j} \quad (3.1)$$

and a probability-weighted value computation

$$v_2^i = \sum_{j=1}^{N_T} p_{T_j}^i \nu_{T_j} \quad (3.2)$$

where  $v^i$  is the value of target  $i$ ,  $p_{T_j}^i$  is the probability that target  $i$  is of type  $T_j$ ,  $\nu_{T_j}$  is the value of target type  $T_j$ , and  $N_T$  is the number of target types. The benefit of the probability-weighted target value computation method is that it allows for a more refined estimate of the target's identification by incorporating the uncertainty in target classification function directly into the target's value. This also permits a more definitive determination of the highest-valued target since the value of individual targets will tend to be more spread out than if the *maximum a posteriori* value computation method was implemented.

Finally, once we have assigned a value to all the targets in the scenario, we have to determine a terminal waypoint for the munition,  $\underline{wp}_{aim}$ . One logical option for computing the aimpoint would be the value-weighted centroid of the targets

$$\underline{wp}_{aim} = \sum_{i=1}^N \tilde{v}^i \widehat{\underline{pos}}_i \quad (3.3)$$

where  $\tilde{v}^i$  is the normalized value for target  $i$  and  $\widehat{\underline{pos}}_i$  is the estimated position for this same target.

### 3.3 Waypoint Generator

This operation establishes the set of candidate paths that is passed to the path selection algorithm. Each path is determined by placing a series of waypoints capable of guiding the munition toward the given termination point. The candidate paths generated in this section are determined in a fashion that guarantees obstacle avoidance, at least in terms of the guidance path. We assume that the vehicle's path planner has knowledge of the location of all obstacles in the scenario, as stated in Chapter 1, and that the obstacles may be represented as simple shapes (primarily rectangular solids). The use of simple shapes is a highly non-restrictive assumption since complex shapes (such as ellipsoids) may be represented by a circumscribed box.

First we must determine if the munition would impact any obstacles while flying the line-of-sight path to the target. This is accomplished by computing the vehicle's position at numerous sample points along this LOS vector and then checking if any position triplet falls within the bounds of the obstacles. If we determine that no impacts occur, then this entire path planning process may be bypassed and the optimal path is found to be simply the LOS path between the munition and the target. On the other hand, for the more important case, in which we must maneuver around an obstacle, we need to place a series of waypoints to guide the munition past the obstacle.

The simple 2-dimension version of this problem tells us to plan three paths. The first path maintains the line-of-sight heading while flying up-and-over the obstacle. The other two paths maintain the current altitude while flying directly around the obstacle. Even though these three paths would be sufficient if we merely want to get past the obstacle, our desire to accomplish this task in an efficient manner drives us to formulate additional path options. For illustrative purposes, we choose to create two more candidate paths somewhere in between the directly over and directly around paths. The obvious option is to fly at a heading halfway between the up-and-over and the around paths while selecting the altitude of the waypoint based on the height of the obstacle. A notional set of waypoints have been placed in Figure 3.2. Note that the paths shown do not actually impact the obstacle, but are offset from the edges by a small safety buffer. Given the type of path

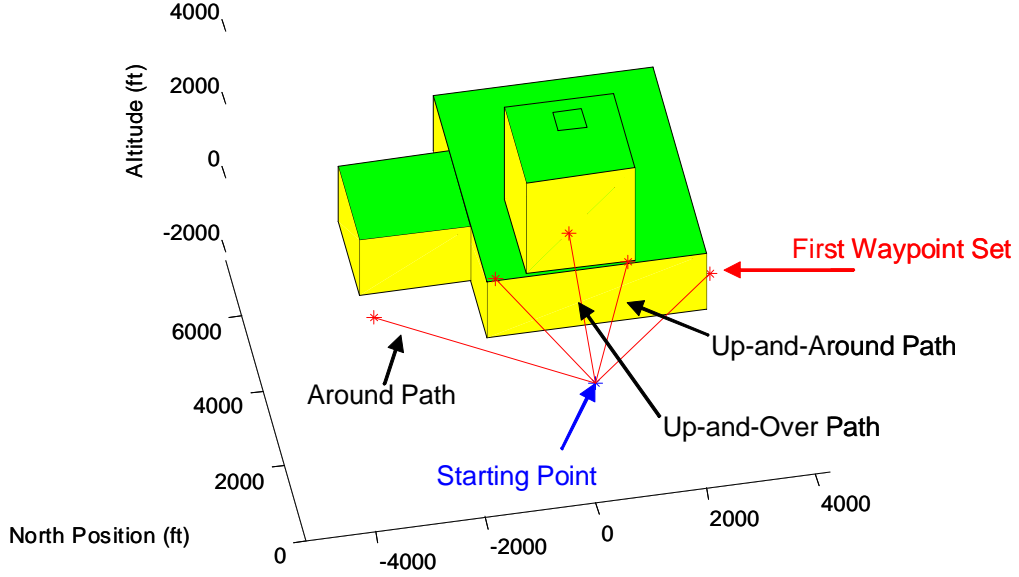


Figure 3.2 Notional waypoint placement - paths do not impact obstacle

(up-and-over, around, up-and-around) and the dimensions of the obstruction, we are now ready to place the waypoints for the 5 candidate paths.

For the simplest path, the up-and-over candidate, we place the altitude of the waypoint at the maximum altitude of the obstacle we are projected to impact and add in a user-defined safety buffer. The north and east position of the waypoint are then fixed to be equal to the location where we initially strike the obstacle. This gives us the first waypoint choice as

$$wp_{up-over} = \begin{bmatrix} p_{north,impact_0} \\ p_{east,impact_0} \\ h_{max} + d_{safety} \end{bmatrix} \quad (3.4)$$

where  $p_{north,impact_0}$  and  $p_{east,impact_0}$  denote the north and east position of the point of impact on the first obstacle we reach,  $h_{max}$  is the maximum altitude of this obstacle, and  $d_{safety}$  is the adjustable safety margin.

For the around paths, our specific placement of waypoints is dependent upon both our general heading and our starting position relative to the obstacle's boundary in the



horizontal plane. For instance, if the heading of the line-of-sight path to the next waypoint is primarily in a northward direction and if the previous waypoint is located to the west of the obstacle, then we should attempt to fly around the obstacle by choosing the northwest and southeast corners of the obstacle as intermediate waypoints. As stated earlier, we maintain the altitude of the previous waypoint for these two new waypoints. A summary of the entire decision logic is shown in Algorithm 1.

In the case of the up-and-around paths, we begin by computing our desired heading,  $\Psi_{up-around}$ , as the angle which bisects the up-and-over and around paths

$$\Psi_{up-around} = \frac{\Psi_{up-over} + \Psi_{around}}{2} \quad (3.5)$$

Next, we determine the vehicle's position at various sample points along this new heading and obtain the same type of obstacle impact information that was used for the up-and-over path. Specifically, we compute the location of first impact and the maximum height of the obstacle. We finish this path type by computing the location of the waypoint in the same manner as Equation (3.4).

Now that we have determined a set of waypoints for the front edge of an obstacle, a series of verifications must be made. In the first verification step, we check to see if the LOS path from the new waypoint to the termination point passes through the same obstacle we are attempting to avoid. If an impact is detected, then we are forced to place an intermediate waypoint based on the type of the path (up-and-over, around, up-and-around) we are extending and the general heading of this LOS path. As an example, if we must extend an up-and-over path and we are heading in a northerly direction, then we shall hold the altitude constant and set the north position of the intermediate waypoint on the northern edge of the obstacle (plus the adjustable safety distance). Although this approach may appear to cause issues in the presence of multi-modal obstacles, the final layer of verification in this series will eliminate this concern. The east position of this waypoint is then computed using simple planar geometry. As shown in Figure 3.3, the east position is given as  $p_{east,new} = p_{east,old} + \Delta_{north} \tan(\Psi)$ . The following three sections of pseudocode present the method used to extend the up-and-over (Algorithm 2),

---

**Algorithm 1** Compute waypoints of "around" type

---

$pos_0$  = position of previous waypoint  
Set altitude of waypoint at altitude of  $pos_0$   
**if** Heading north **then**  
    **if**  $pos_0$  is west of obstacle **then**  
5:     Place waypoints on northwest and southeast corners of obstacle  
    **else if**  $pos_0$  is east of obstacle **then**  
        Place waypoints on northeast and southwest corners of obstacle  
    **else**  
        Place waypoints on southwest and southeast corners of obstacle  
10:    **end if**  
    **else if** Heading south **then**  
        **if**  $pos_0$  is west of obstacle **then**  
            Place waypoints on southwest and northeast corners of obstacle  
        **else if**  $pos_0$  is east of obstacle **then**  
15:     Place waypoints on northwest and southeast corners of obstacle  
        **else**  
            Place waypoints on northwest and northeast corners of obstacle  
        **end if**  
    **else if** Heading east **then**  
20:    **if**  $pos_0$  is north of obstacle **then**  
            Place waypoints on northeast and southwest corners of obstacle  
        **else if**  $pos_0$  is south of obstacle **then**  
            Place waypoints on northwest and southeast corners of obstacle  
        **else**  
25:     Place waypoints on northwest and southwest corners of obstacle  
        **end if**  
    **else**  
        **if**  $pos_0$  is north of obstacle **then**  
            Place waypoints on northwest and southeast corners of obstacle  
30:    **else if**  $pos_0$  is south of obstacle **then**  
            Place waypoints on northeast and southwest corners of obstacle  
        **else**  
            Place waypoints on northeast and southeast corners of obstacle  
        **end if**  
35: **end if**

---

around (Algorithm 3), and up-and-around (Algorithm 4) waypoints. The extension of the waypoints shown in Figure 3.2 is seen in Figure 3.4.

---

**Algorithm 2** Extending "up-and-over" paths

---

```

 $pos_0$  = position of previous waypoint
 $\Psi_0$  = heading of path at  $pos_0$ 
Set altitude of waypoint at altitude of  $pos_0$ 
if Heading north then
5:   Follow  $\Psi_0$  to north edge of obstacle
else if Heading south then
   Follow  $\Psi_0$  to south edge of obstacle
else if Heading east then
   Follow  $\Psi_0$  to east edge of obstacle
10: else
   Follow  $\Psi_0$  to west edge of obstacle
end if

```

---



---

**Algorithm 3** Extending "around" paths

---

```

 $pos_0$  = position of previous waypoint
Set altitude of waypoint at altitude of  $pos_0$ 
if Heading north then
   Extend waypoints to northern corners of obstacle
5: else if Heading south then
   Extend waypoints to southern corners of obstacle
else if Heading east then
   Extend waypoints to eastern corners of obstacle
else
10: Extend waypoints to western corners of obstacle
end if

```

---

The second level of verification is only invoked if an intermediate waypoint was generated as a result of the first layer of checks. At this point we are still concerned with impacting the same obstacle, even though we do not expect there to be any further trouble with this obstacle. If an obstacle collision is detected on the LOS path between the intermediate waypoint and the waypoint from which it was extended, then this entire candidate path may be discarded as a dead-end.

The final verification layer, which is always necessary, checks for obstacle crossings on the LOS between the most recently generated waypoint and the terminal waypoint. If no collision is detected, then this candidate path is complete. On the other hand, when

---

**Algorithm 4** Extending "up-and-around" paths

---

$pos_0$  = position of previous waypoint  
 $\gamma_{path,0}$  = flight path angle at  $pos_0$   
 $\Psi_0$  = heading angle at  $pos_0$   
**if** Heading north **then**  
5:   Extend waypoints along  $\Psi_0$  and  $\gamma_{path,0}$  to northern edge of obstacle  
**else if** Heading south **then**  
    Extend waypoints along  $\Psi_0$  and  $\gamma_{path,0}$  to southern edge of obstacle  
**else if** Heading east **then**  
    Extend waypoints along  $\Psi_0$  and  $\gamma_{path,0}$  to eastern edge of obstacle  
10: **else**  
    Extend waypoints along  $\Psi_0$  and  $\gamma_{path,0}$  to western edge of obstacle  
**end if**

---

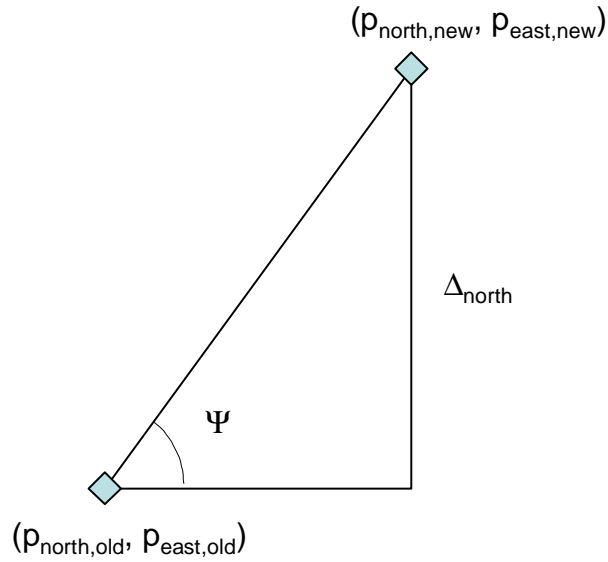


Figure 3.3   Waypoint extension

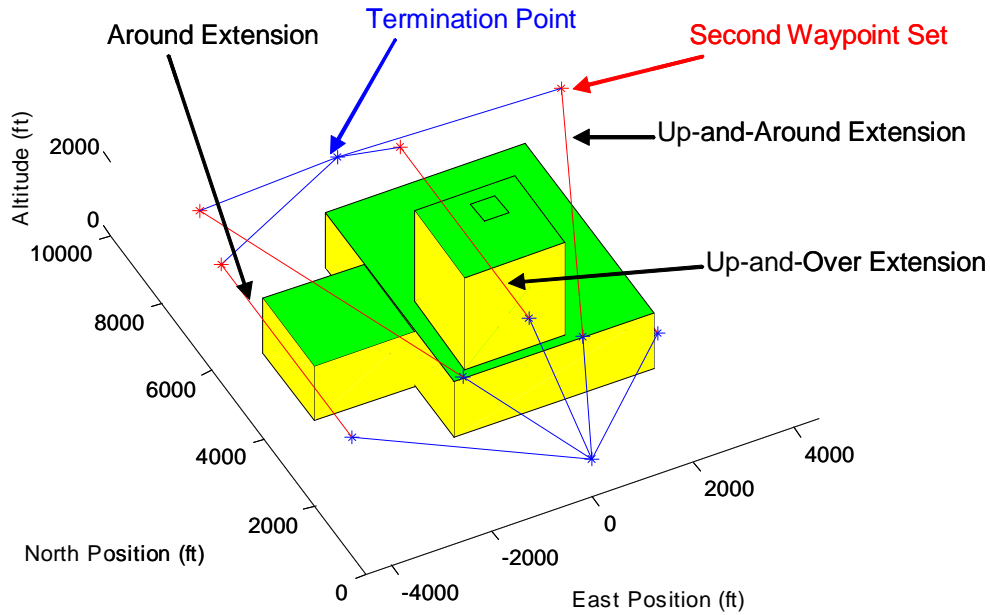


Figure 3.4 Extension of notional waypoints

a collision with a different obstacle is detected, the waypoint generation process must be repeated using the most recent waypoint as the starting position.

This entire process is repeated until all candidate paths are designated as either completed or dead-ends. As one can see, this algorithm has the potential of generating up to  $5^{n_{obst}}$  candidate paths, where  $n_{obst}$  is the number of obstacles between the munition's launch point and the target's location. The full waypoint generation process is summarized in Algorithm 5.

### 3.4 Path Selection

The role of the path selection block in Figure 3.1 is to determine the optimal obstacle-free path from a given set of potential flight paths. Minimum effort control theory provides the basis for making this determination. This algorithm is also capable of identifying paths that are not feasible to traverse for the set of given flight conditions. A cost, made up of three components, is computed for each candidate path and has the following general

---

**Algorithm 5** Top-level Waypoint Generation

---

```
    Check for obstacle collisions
    if collision detected then
        Generate set of 5 waypoints
        for all new waypoints do
5:     Check for obstacle collisions on same obstacle along LOS path to aimpoint
        if collision detected then
            Extend the waypoint
            Check for obstacle collisions on same obstacle along LOS path from previous
            waypoint
            if collision detected then
10:         Delete candidate path
            end if
        end if
        Check for obstacle collisions on new obstacle along LOS path to aimpoint
        if collision detected then
15:         Initiate new instance of waypoint generation
        else
            Declare candidate path complete
        end if
    end for
20: end if
```

---

form:

$$J_{path} = J_{orientation} + J_{travel} + J_{distance} \quad (3.6)$$

The explanation of each component of this cost will be given later in this section. As an innovation of this research, these component costs are chosen based on the way the problem is decomposed. Additionally, early test cases indicated that the north and east position states from the state vector of Equation (2.48) led to an unstable system. Since the vehicle is assumed to fly at a constant speed, one cannot expect the munition to reach a steady-state value for north and east position. The data provided by these two states must be ignored for the purpose of computing controllability Grammians. Given this loss of important information,  $J_{distance}$  is an effort to address the fact that a portion of the expended energy will be proportional to the total distance travelled. The candidate path with the lowest total cost is then selected as the desired set of waypoints for the munition to follow in order to reach the target. We will now outline the general procedure to be

followed in computing the *orientation* and *travel* costs, starting with a brief discussion of the controllability Grammian concept which is shared by both of these component costs.

First, a linearized system model of the air vehicle itself is generated. This linearized state-space model is obtained by numerically computing the Jacobian matrix [11] for the six-degree-of-freedom F-16 aircraft plant model given in Figure 2.5 and evaluating this Jacobian at a given set of trimmed flight conditions. These trim conditions are assumed to be the equilibrium state values and the equilibrium input values required to fly the vehicle in a specified manner, such as steady-level flight or wings-level nose-up (or nose-down) flight. Next, the feedback control loops are closed in order to form the full linearized state-space model of the system under consideration. With this linearized model available, we finally compute the steady-state controllability Grammian using Equation (2.43) for the total system under the current equilibrium state and input conditions. A cost may now be computed as the norm squared value of the minimum energy control, in a manner similar to Equation (2.45), required to reach a certain set of state values

$$\|\underline{u}^*\|^2 = \sum_{i=1}^n \frac{\alpha_i^2}{\lambda_i} \quad (3.7)$$

where  $\lambda_i$  is the  $i^{th}$  eigenvalue of the controllability Grammian and  $\alpha_i$  is a scale factor associated with this eigenvalue. Note that this is an extension of Equation (2.45) to the case of a more generalized desired state vector.

Before we continue to detail the unique steps taken to compute each component cost, we must give a proof of how this generic cost is computed from the controllability Grammian.

**Proof.** Let us assume that we have a stable system described in state-space by the matrices  $A$ ,  $B$ ,  $C$ , and  $D$  (Equation (2.36)). Since the system is stable, we are guaranteed that the steady-state controllability Grammian,  $W_{css}$ , can be obtained by solving Equation (2.43). Furthermore, let us also assume that  $W_{css}$  is of full rank so that the system is completely controllable (property 3 of controllability Grammians). This last assumption is not necessary but will make the proof a bit easier. Later in the path selection process,

a modification of this assumption will be used as a built-in method to reduce the number of candidate paths.

To get the proof started, we simply compute the eigenvalues and unit-norm eigenvectors of  $W_{css}$ . From linear algebra, we know that the eigenvalues for a real, symmetric matrix are guaranteed to be real and that the eigenvectors for distinct eigenvalues will be orthogonal to each other. Now, for any desired state  $\underline{x}$ , we evaluate the scalar projection of  $\underline{x}$  along each  $\underline{\xi}_i$  to obtain a set of scale factors which describe the amount of state motion required in the  $\underline{\xi}_i$  direction of state-space

$$\alpha_i = \underline{x}^T \underline{\xi}_i \quad (3.8)$$

Given that  $W_{css}$  is real and symmetric and assuming that the  $n$  eigenvalues are all distinct (this is not an unrealistic assumption), the  $n$  eigenvectors are known to be linearly independent of each other and we may write the desired state vector as

$$\underline{x} = \sum_{i=1}^n \alpha_i \underline{\xi}_i \quad (3.9)$$

Next, we solve for the energy of the optimal control input vector using the following norm expression

$$\|\underline{u}^*\|^2 \triangleq \int_0^{t_f} \underline{u}^{*T}(\tau) \underline{u}^*(\tau) d\tau \quad (3.10)$$

After substituting in the form of the control input from Equation (2.47), the energy of this vector is given by

$$\|\underline{u}^*\|^2 = \underline{x}^T W_{css}^{-1} [\Phi(t_f) \int_0^{t_f} \Phi(-\tau) B B^T \Phi^T(-\tau) d\tau \Phi^T(t_f)] W_{css}^{-1} \underline{x} \quad (3.11)$$

where we have applied the properties of the state transition matrix such that  $\Phi^T(t_f - \tau) = \Phi^T(-\tau) \Phi^T(t_f)$ . The bracketed term in Equation (3.11) is the steady-state controllability Grammian by definition (Equation (2.41)). Replacing  $\underline{x}$  with the form of Equation (3.9),



the norm squared value of the input can now be reduced to

$$\|\underline{u}^*\|^2 = \left( \sum_{i=1}^n \alpha_i \xi_{\underline{i}}^T \right) W_{c_{ss}}^{-1} \left( \sum_{j=1}^n \alpha_j \xi_{\underline{j}} \right) \quad (3.12)$$

We know that the eigenvalues and eigenvectors for a given matrix ( $\Omega$ ) are related to each other by  $\Omega \xi_{\underline{j}} = \lambda_j \xi_{\underline{j}}$  [18] and it is easy to see that  $\frac{1}{\lambda_j} \xi_{\underline{j}} = \Omega^{-1} \xi_{\underline{j}}$ , as long as the inverse of  $\Omega$  exists. Applying this to our input energy computation, the energy of the input becomes

$$\|\underline{u}^*\|^2 = \left( \sum_{i=1}^n \alpha_i \xi_{\underline{i}}^T \right) \left( \sum_{j=1}^n \frac{\alpha_j}{\lambda_j} \xi_{\underline{j}} \right) \quad (3.13)$$

Since the eigenvectors are presumed to all be mutually orthogonal, we may merge the summations to obtain

$$\|\underline{u}^*\|^2 = \sum_{i=1}^n \frac{\alpha_i^2}{\lambda_i} \xi_{\underline{i}}^T \xi_{\underline{i}} \quad (3.14)$$

Noting that the eigenvectors used in this problem are forced to be of unit length ( $\xi_{\underline{i}}^T \xi_{\underline{i}} = 1$ ), the norm squared value of the minimum energy control required to reach the desired state is therefore given by Equation (3.7). ■

It must be noted here that the  $\underline{x}$  vector used throughout the preceding proof is really the desired change in system state such that, for a system linearized about some nominal state

$$\underline{x} = \underline{x}_{desired} - \underline{x}_{nominal} \quad (3.15)$$

We are now ready to deal with the specifics of each component cost.

For the *orientation* cost, we are primarily concerned with the energy required to make changes in the vehicle's heading and flight path angles. However, given that the angle of attack and pitch angle will vary slightly as a function of altitude, this component cost will also consider the energy needed to achieve an altitude change. These energies will be referred to, respectively, as the orientation energy ( $\|\underline{u}^*\|_{orient}^2$ ) and the height energy ( $\|\underline{u}^*\|_{height}^2$ ), and we will consider each in turn.

In the case of the orientation energy, the computed equilibrium state and input values, to be used in the system linearization process, correspond to the air vehicle's flight parameters (speed, altitude, and flight path angle) at the current waypoint. The state at which we desire arrive is identical to the current equilibrium state except that the heading and flight path angle information is replaced by the angles required to turn toward the next waypoint. For the height energy, the starting equilibrium conditions are the state and input values to which we drove the system while computing the orientation cost. The goal state for this computation is merely the current equilibrium state with the altitude of the next waypoint replacing that of the current waypoint. The overall *orientation* cost for the  $l^{th}$  candidate path is given by

$$J_{orientation,l} = \|\underline{u}^*\|_{orient,l}^2 + \|\underline{u}^*\|_{height,l}^2 \quad (3.16)$$

or in a generalized form which allows for relative weighting between the orientation and height energies

$$J_{orientation,l} = W_1 \|\underline{u}^*\|_{orient,l}^2 + W_2 \|\underline{u}^*\|_{height,l}^2 \quad (3.17)$$

For the *travel* cost, we are interested in addressing the energy involved in merely flying on a straight and level path. Bearing this in mind, the equilibrium values used in the system linearization are computed based on steady-level flight at a constant speed and at the altitude of the next waypoint. Due to our desire to fly straight and level over a given distance ( $d$ ), we must alter the energy computation to account for this distance traveled. In order to keep the computation balanced correctly, we must also modify the state vector we are driving toward. After incorporating these changes, we arrive at the following equations to be used only for the travel cost computations

$$\underline{x} = \sum_{i=1}^n \frac{\alpha_i \underline{\xi}_i}{d} \quad (3.18)$$

instead of Equation (3.9) and

$$\|\underline{u}^*\|_{travel}^2 = d^2 \sum_{i=1}^n \frac{\alpha_i^2}{\lambda_i} \quad (3.19)$$

instead of Equation (3.7). The overall *travel* cost for the  $l^{th}$  candidate path is given by

$$J_{travel,l} = \|\underline{u}^*\|_{travel,l}^2 \quad (3.20)$$

The *distance* cost, although it is computed based on the distance travelled, is really misnamed. Since we have made the assumption that the vehicle maintains a given speed throughout its flight, this is really a manner of incurring a cost for prolonged flight times. The rationale behind this cost component is that, lacking an explicit expression for fuel consumption, a longer flight will tend to require more propellant than a shorter flight. This cost is based on calculating the total distance ( $dist_{tot,l}$ ) traveled for the  $l^{th}$  candidate path as shown below

$$dist_{tot,l} = \sum_{i=1}^{n_{wp}-1} \left\| \underline{wp}_{i+1} - \underline{wp}_i \right\| \quad (3.21)$$

where  $n_{wp}$  is the number of waypoints on the path under consideration

The procedure described above is used to compute the cost to fly between any two waypoints in a candidate path. Once the energies and distances are fully computed for each candidate path, we are finally ready to determine the component costs associated with each path. In order to ensure that the values are all around the same rough order of magnitude, we will simply normalize the data in each component, so that we arrive at the following component costs

$$\begin{aligned} \tilde{J}_{orientation,l} &= \frac{J_{orientation,l}}{\sum_{i=1}^{n_{paths}} J_{orientation,i}} \\ \tilde{J}_{travel,l} &= \frac{J_{travel,l}}{\sum_{i=1}^{n_{paths}} J_{travel,i}} \\ \tilde{J}_{distance,l} &= \frac{dist_{tot,l}}{\sum_{i=1}^{n_{paths}} dist_{tot,i}} \end{aligned} \quad (3.22)$$

where  $n_{paths}$  is the total number of candidate paths being evaluated. The net effect of this normalization is to give each cost component an equal weighting in the overall cost computation

$$J_{path,l} = \tilde{J}_{orientation,l} + \tilde{J}_{travel,l} + \tilde{J}_{distance,l} \quad (3.23)$$

One could readily alter the relative weighting of these costs by including a multiplicative factor on each component cost

$$J_{path,l} = W_3 \tilde{J}_{orientation,l} + W_4 \tilde{J}_{travel,l} + W_5 \tilde{J}_{distance,l} \quad (3.24)$$

This cost with relative weighting factors is a generalization of Equation (3.6) and exemplifies the fact that we have given each component cost an equal vote in determining the total path cost.

If, at any point in the procedure delineated above, the linearized system is found to be unstable, then the entire candidate path under consideration at that instant may immediately be discarded. In addition, if the steady-state controllability Grammian indicates that the system is not completely controllable, as evidenced by a zero eigenvalue, and we desire the system to move in an uncontrollable direction of state-space, as seen by a non-zero  $\alpha_i$  value corresponding to the zero eigenvalue, then this candidate path may be ruled out based on the infeasibility of arriving at one of the waypoints. This entire path selection procedure is summarized in Algorithm 6.

---

**Algorithm 6** Path Selection Algorithm

---

```

for  $i = 1$  to number of candidate paths do
  if  $candidate_i$  exists then
    for  $j = 1$  to number of waypoints in  $candidate_i - 1$  do
      if system unstable at waypoint  $j$  or waypoint  $j$  is in uncontrollable space then
5:      Ignore  $candidate_i$ 
    else
      Compute cumulative energy for orientation and height changes for  $candidate_i$ 
      Compute cumulative energy to travel on  $candidate_i$ 
      Compute cumulative distance travelled on  $candidate_i$ 
10:    end if
    end for
  end if
  end for
  Normalize orientation energy, travel energy, and distance for each candidate path
15: Compute total cost for each candidate path
  Select path with minimum cost

```

---

### 3.5 Command Generator

The function of this block is to generate the optimal input which will drive the vehicle to the next waypoint under the assumption that we wish to fly along the line-of-sight path as much as possible. Though there are numerous optimal control methods, this research is concerned with extending the work on CLC [1] and MPC [9] control schemes to 3-dimensional problems and presenting a minimum effort control technique which makes use of data obtained during the path selection procedure. While all three of these control optimization approaches require the specification of a set of waypoints, only MPC imposes the additional need to generate a reference path between the waypoints. The general procedure for these three command generation approaches is described here and the details of a MATLAB<sup>®</sup> implementation of the MPC and minimum effort control techniques will be given in Chapter 4.

*3.5.1 Extremal Input Commanding.* Recall that within the CLC optimal control technique is the desire to minimize the deviation from the line-of-sight path between waypoints while operating under the assumption that all turns are made at the vehicle's maximum rate of turn. In fact, this control optimization method does not directly compute any control inputs; instead the CLC method determines the *time* to apply one of a given set of inputs. The specific control inputs that make up this set of potential inputs are completely known *a priori*, and the chosen input is based on which direction the vehicle is attempting to turn. However, this simple approach is only valid for a 2-dimensional problem. In this section, we present the extension to a 3-dimensional scenario and attempt to show where this process breaks down.

First, we extend the fundamental structure of the work in [1] to the full 3-dimensional case, in which the position triplet has x, y, and z directions and the control inputs are heading ( $\Psi$ ) and flight path ( $\gamma_{path}$ ) angles. Within this framework, the system description

of Equation (2.21) becomes

$$\begin{aligned}
\dot{x}(t) &= s \cos(\Psi(t)) \cos(\gamma_{path}(t)) \\
\dot{y}(t) &= s \sin(\Psi(t)) \cos(\gamma_{path}(t)) \\
\dot{z}(t) &= s \sin(\gamma_{path}(t)) \\
\dot{\Psi}(t) &= u_1(t) \\
\dot{\gamma}_{path}(t) &= u_2(t)
\end{aligned} \tag{3.25}$$

and the maximum control input inequality of Equation (2.22) is expanded to

$$\begin{aligned}
\|u_1\| &\leq u_{1,max} \\
\|u_2\| &\leq u_{2,max}
\end{aligned} \tag{3.26}$$

However, this set of constraints on the control input is incomplete. In any realizable air vehicle, there exists a necessary trade-off between turning and climbing performance. For instance, this relationship may be expressed in terms of a maximum centripetal acceleration, such that the application of both control inputs may not exceed a certain  $g$ -load threshold. Without presenting a form for the exact relationship, we are assured of the fact that one cannot blindly attempt to apply the maximum control input on both input channels. Keeping in mind that we are operating with an incomplete set of control input constraints, we will continue this 3-dimensional extension until the need for additional system characterization is more apparent.

Our next step is to determine the 3-dimensional version of the system Hamiltonian

$$\begin{aligned}
H(\underline{x}(t), \underline{u}(t), \gamma_0, \underline{\gamma}(t)) &= \gamma_0 + [\gamma_1(t)s \cos(\Psi(t)) + \gamma_2(t)s \sin(\Psi(t))] \cos(\gamma_{path}(t)) \\
&\quad + \gamma_3(t)s \sin(\gamma_{path}(t)) + \gamma_4(t)u_1(t) + \gamma_5(t)u_2(t)
\end{aligned} \tag{3.27}$$

and then begin to apply the conditions of Pontryagin's minimum principle from Equation (2.18) and the auxiliary inequality condition implied by Equation (2.20). Avoiding the exercise in simple mathematics, we concentrate on the inequality shown below since it

provides an excellent indication of insufficient system information

$$H(\underline{x}(t), \underline{u}^*(t), \gamma_0, \underline{\gamma}(t)) \leq H(\underline{x}(t), \underline{u}(t), \gamma_0, \underline{\gamma}(t)), \quad \forall \underline{u} \in \mu \quad (3.28)$$

After inserting the Hamiltonian of Equation (3.27), using the same cost function as in Equation (2.23), and simplifying the expressions, we arrive at the following equation

$$\gamma_4(t)u_1^*(t) + \gamma_5(t)u_2^*(t) \leq \gamma_4(t)u_1(t) + \gamma_5(t)u_2(t), \quad \forall \underline{u} \in \mu \quad (3.29)$$

Here is a prime example of the need for a manner to express the trade-off between the two control inputs. Unlike the results from Equation (2.25) and Equation (2.26), the relation above does not have a simple specific solution.

At this point, the 3-dimensional extension of the CLC control technique must be deferred until an adequate characterization of the interplay between heading rate and climb rate is available and a recommendation will be presented in Chapter 5.

*3.5.2 Model Predictive Control Commanding.* Conceptually, the extension of model predictive control techniques to a 3-dimensional approach is quite easy. The form of the non-linear system model of Equation (2.14) and the MPC cost summation of Equation (2.31) is unaltered. We are able to apply the same general procedure, as outlined for 2-dimensional problems in [9], to determine the optimal *perturbation* control input. However, the complexity of the problem is greatly increased by the addition of a third direction of motion. In order to complete this 3-dimension extension of the MPC scheme, two modifications to the problem description are required.

The first change that must be made is to define the six-degree-of-freedom flight model in the form of the general non-linear system of Equation (2.14). Let us assume the state

and control input vectors are defined for a 2-dimensional world as

$$\underline{x} = \begin{bmatrix} V \\ p_{north} \\ p_{east} \\ \psi \\ \psi_{rate} \end{bmatrix}, \quad \underline{u} = \begin{bmatrix} throttle \\ rdr_{cmd} \end{bmatrix} \quad (3.30)$$

where  $V$  is the magnitude of the vehicle's velocity vector,  $p_{north}$  and  $p_{east}$  are the northward and eastward positions of the vehicle,  $\psi$  is the yaw angle, and  $\psi_{rate}$  is the rate of change of the yaw angle. Note that the *throttle* input is a percentage of the maximum throttle setting and the  $rdr_{cmd}$  input is the commanded rudder deflection angle. In order to extend this problem to a 3-dimensional world, the state vector must be augmented to incorporate the additional states which account for the linear and rotational kinematics in the new degrees of freedom, and the control input vector must be expanded to include commands for the remaining control surfaces. This augmentation process modifies the state and control input vectors to their generic 3-dimensional definitions

$$\underline{x} = \begin{bmatrix} V \\ p_{north} \\ p_{east} \\ h \\ \phi \\ \theta \\ \psi \\ \phi_{rate} \\ \theta_{rate} \\ \psi_{rate} \end{bmatrix}, \quad \underline{u} = \begin{bmatrix} throttle \\ ail_{cmd} \\ ele_{cmd} \\ rdr_{cmd} \end{bmatrix} \quad (3.31)$$

where  $h$  is the vehicle's altitude,  $\phi$  is the roll angle,  $\theta$  is the pitch angle, and  $\phi_{rate}$  and  $\theta_{rate}$  are the rates of change of the roll and pitch angles, respectively. Similar to the control input vector of Equation (3.30),  $ail_{cmd}$  is the commanded aileron deflection angle



and  $ele_{cmd}$  is the commanded elevator deflection angle. The states defined in Equation (3.31) are not the full set of states that would typically be included for a realistic model (additional states may be required based on the specific controller design), but this state vector definition illustrates the simple fact that the expansion to a fully 3-dimensional scenario entails a significant increase in the amount of data that must be handled. This increased complexity propagates itself throughout the MPC structure and generates a much more computationally burdensome optimization problem.

The secondary, yet equally critical alteration that must occur is to expand the reference trajectory to incorporate all three dimensions of the vehicle's position. Although both changes are necessary, this modification is what really provides the capability of MPC to be exploited for 3-dimensional control optimization problems. The waypoints that define the guidance path chosen by Algorithm 6 during the path selection operation in Figure 3.1 provide the endpoints for each segment of the reference trajectory, and one must then interpolate between the points to determine an appropriate reference trajectory. One obvious interpolation technique is simply to project the line-of-sight between successive waypoints. Given the method used to generate the set of waypoints, this LOS projection guarantees that the reference trajectory will not pass through any of the obstacles.

Despite our best efforts, a properly functioning version of the MPC-based optimization algorithm was not achieved. Nevertheless, we were able to gain some insight into possible avenues of investigation for this control generation approach. These recommendations will be detailed in Chapter 5.

*3.5.3 Minimum Effort Control Commanding.* In the typical minimum effort control setting, the controllability Grammian is used to determine the full set of inputs which will affect the desired state change while requiring the least amount of energy at the input. Note that this is not a perturbation control method. These control inputs are specified by Equation (2.47), which is reprinted below

$$\underline{u}^*(t) = B^T \Phi^T(t_f - t) W_c^{-1}(t_f) \underline{x}, \quad 0 < t \leq t_f$$

Although this basic control optimization technique may be used to guide a system to a specific set of points in state-space, there is no method to predetermine the path that the state trajectory will follow in between the state vector setpoints. The state trajectory resulting from the application of a control input generated by minimum effort control methods may "wander" through state-space before reaching the terminal point. Given that we desire to avoid obstacles, a specific "safe" path must be followed through state-space and a modified view of minimum effort control is necessary in order to make this technique a viable option.

This "safe" trajectory has already been determined, as the line-of-sight path between waypoints, by the waypoint generation and path selection processes. Furthermore, the optimal set of waypoints was selected by using the controllability Grammian in a manner based on minimum effort control. Recalling that one of the initial assumptions of this research is that heading, altitude, and velocity hold autopilots are available, the information explicitly contained within the chosen set of waypoints may be used to generate the control inputs directly. In order to travel between two consecutive waypoints,  $\underline{wp}_{current}$  and  $\underline{wp}_{next}$ , the specific inputs applied to the system are the desired speed, the altitude of  $\underline{wp}_{next}$ , and the horizontal plane heading angle of the vector from  $\underline{wp}_{current}$  to  $\underline{wp}_{next}$ . Finally, to ensure that the line-of-sight path is closely followed, the control inputs must be generated at a sufficiently high enough rate. Bearing this in mind, the series of inputs are computed such that all turns and altitude changes are accomplished in a smooth manner and the line-of-sight path is closely followed.

### 3.6 *Flight Simulator*

Based on our assumptions and the data available from the command generator, a few modifications to the basic simulation model of Figure 2.5 are necessary. The first change is merely the removal of the linear ( $\text{lin\_accel}$ ) and rotational ( $\text{rot\_accel}$ ) acceleration terms. Recall that, in the original project for which this model was used [11], these linear and rotational acceleration terms were included in an effort to evaluate the forces sensed by a pilot in various flight conditions and were not used to modify the dynamics of the overall system. The omission of these six terms is possible since we are not interested in this

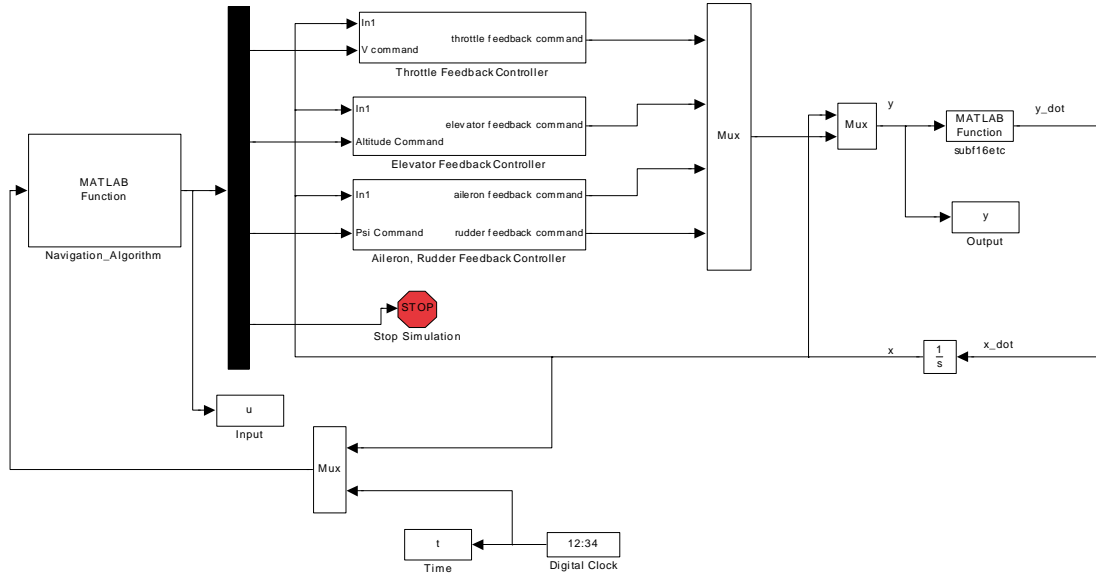


Figure 3.5 Modified simulation model

information and the loss of this data does not degrade the performance of the rest of the system. Another simple change is the addition of an exit condition to halt the simulation prematurely when the munition reaches the intended target.

The more significant alterations are in the specific input we apply to the system and the details of the feedback controllers. In general, the input we apply is determined by the control optimization method we choose to employ, and this choice will force changes to the feedback controller structure. Since the modified minimum effort control technique is the only command generation method being implemented in three dimensions, the control input takes the form of a commanded speed, altitude, and heading. While the controllers are still autopilots designed to hold the vehicle in a given configuration, we have redefined the reference values to be the desired speed, altitude and heading, instead of the equilibrium conditions. This change is required since the original model of Figure 2.5 was designed to track the equilibrium speed, altitude, and heading values and external inputs were available to apply changes directly to the actuator deflection angles. For simplicity, these autopilots are designed as simple proportional feedback controllers which have constant gains determined in an *ad hoc* manner and provide an adequate level of responsiveness. The new simulation model is shown in Figure 3.5.

### *3.7 Summary*

In this chapter, we have presented the various innovations of this thesis. Section 3.2 detailed a simple, intuitive method to assess the value of a given target within a multi-target scenario. Next, Section 3.3 developed the procedure to determine potential waypoints for the purpose of obstacle avoidance. Section 3.4 then employed minimum effort control concepts to derive a method to select a desired path from the candidate trajectories defined during the waypoint generation process. The extension of CLC, MPC, and minimum effort control techniques to three dimensions was presented in Section 3.5. However, functioning versions of the CLC and MPC methods were not achieved. Finally, Section 3.6 detailed the adaptation of the simulation model to account for the unique elements of this research. Given the algorithms and simulation framework developed in this chapter, we are ready to discuss the test cases and analyze the results of these simulations.

## *IV. Simulation Implementation and Results*

### *4.1 Introduction*

Throughout this chapter, we present several test cases in an effort to highlight both the strengths and weaknesses of the path planning and control optimization procedures. In general, two planned paths are illustrated for each scenario. The first path, resulting from the path selection algorithm developed in Section 3.4 of this research, is known as the Minimum Effort Path (MEP). The other path is the Minimum Distance Path (MDP), which is obtained by selecting the path with the lowest cumulative distance. Once a candidate path is selected, a command generator function is used to compute the control inputs and thus drive the vehicle along the selected path. The MPC algorithm was originally chosen to perform this function, but due to implementation issues with the MPC approach, a simplified command generator based on the MEP waypoints is used. As such, one can envision the problem being decomposed into navigation and control functions, where the MEP and MDP selection processes represent navigation. The control methods include the CLC, MPC, and MEP-based command generator techniques. As stated in Chapter 3, the extremal control (CLC) and model predictive control optimization approaches are not used in these simulations, but we will discuss the implementation issues encountered when using the MPC command generator.

We start the analysis with a description of the MATLAB<sup>®</sup> implementation of the MPC and MEP-based command generator techniques. Following the discussion of implementation issues, two basic scenarios are presented to demonstrate the simulation environment. Next, scenarios of the first test set are designed to highlight the characteristics of the path planning algorithm. Finally, the simulations of the second test set are devoted to the characteristics of the MEP-based control technique. Note that, for now, we assume the targets are stationary and the target tracking aspect is bypassed.

## 4.2 Command Generators

As mentioned above, efforts were made to implement the MPC and MEP-based command generators in MATLAB<sup>®</sup>. The following is a presentation of the implementation details and a discussion of the issues which arose from each command generation technique.

*4.2.1 Model Predictive Control Command Generator.* As stated at the end of Section 3.5.2, a functioning version of the 3-dimensional MPC optimization technique was not achieved. However, the important aspects of the implementation of this approach in MATLAB<sup>®</sup> must be noted in order to assist any future efforts related to this topic. Recalling that the general form of this method in three dimensions is identical to the form of the 2-dimensional problem, we revisit the presentation of the MPC approach in Section 2.3.2. Specifically, we focus on Equation (2.33), which is reproduced below

$$J_k = \underline{\eta}^T (Y^T \Gamma Y + U^T \Upsilon U) \underline{\eta} + 2[(\underline{y}_{nom} - \underline{r})^T \Gamma Y + \underline{u}_{nom}^T \Upsilon U] \underline{\eta}$$

and explain the method to obtain the  $U$  and  $Y$  matrices.

First, the simpler of the two matrices,  $U$ , is formed for a single input by selecting a set of basis functions. The basis functions are then evaluated at  $N$  discrete-time sample points, where  $N$  is the length of the finite horizon. If  $T_h$  is the continuous-time interval for the finite horizon and  $T_s$  is the sample interval, then the exact discrete-time interval is computed as  $N = \frac{T_h}{T_s}$ . Now, for each basis function, we obtain a vector of  $N$  values and, given that we intend to implement this technique with several basis functions and multiple inputs, we orient the vector so that the dimensions are  $1 \times 1 \times N$ . For a set of  $n_{basis}$  basis functions, the  $U$  matrix is formed such that the individual vectors constitute the columns of the matrix and the dimension of  $U$  is  $1 \times n_{basis} \times N$ . When we extend this to the general case of  $m$  inputs, the final  $U$  matrix is generated by placing identical copies of the single-input form of  $U$  on the main diagonal of a block diagonal matrix and the dimensions of this final matrix are  $m \times (m * n_{basis}) \times N$ . As an example, for the four input system defined by Equation (3.31) and a set of five Laguerre polynomial basis functions,

the single-input version of the  $U$  matrix is given by

$$U_1 = \begin{bmatrix} \varepsilon_0 & \varepsilon_1 & \varepsilon_2 & \varepsilon_3 & \varepsilon_4 \end{bmatrix} \quad (4.1)$$

where  $\varepsilon_i$  is computed using [16]

$$\varepsilon_i(k) = \frac{e^{kT_s}}{i!} \frac{d^i[(kT_s)^i e^{-kT_s}]}{dt^i}, \quad i = 0 : 4, \quad k = 0 : (N - 1) \quad (4.2)$$

Note that these  $\varepsilon$  functions are determined by the choice of the basis functions. Finally, the multi-input version is formulated as

$$U = \begin{bmatrix} U_1 & 0_{1 \times 5} & 0_{1 \times 5} & 0_{1 \times 5} \\ 0_{1 \times 5} & U_1 & 0_{1 \times 5} & 0_{1 \times 5} \\ 0_{1 \times 5} & 0_{1 \times 5} & U_1 & 0_{1 \times 5} \\ 0_{1 \times 5} & 0_{1 \times 5} & 0_{1 \times 5} & U_1 \end{bmatrix} \quad (4.3)$$

where  $0_{1 \times 5}$  is a row vector of five zeros and the full dimension of  $U$  is  $4 \times 20 \times N$ .

Next, we consider the more complex generation of the output relationship matrix,  $Y$ . In Chapter 2 we mentioned that  $Y$  is formed by exciting the nonlinear system with a series of test inputs on a channel by channel basis. Now we explain this process in greater detail, starting with the formulation of the input to be applied. Since we are working with a *set* of basis functions to represent the perturbations about some nominal input, we must determine the system response to each basis function as subdivisions of the response due to input channelization. In order to accomplish this task, we form the perturbed input as

$$u_i^l(k) = u_0^l(k) + U(k)\underline{\eta}_i \quad (4.4)$$

where  $u_i^l(k)$  is the  $l^{th}$  input perturbed by the  $i^{th}$  basis function at time sample  $k$ ,  $u_0^l(k)$  is the  $l^{th}$  nominal input at time sample  $k$ ,  $U(k)$  is the  $U$  matrix at time sample  $k$ , and  $\underline{\eta}_i$  is used to isolate the  $i^{th}$  basis function. For example, to isolate the third function in a set

of five basis functions, we use

$$\underline{\eta}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T \quad (4.5)$$

The nominal input used throughout this research is the equilibrium input values required for the munition to maintain a given set of trim conditions (speed, heading, and flight-path angle). In order to determine the system response to the perturbational component of the input over the finite horizon, we must first excite the nonlinear system with the perturbed input and then subtract off the nominal system response. The nominal system response is obtained by driving the nonlinear system with the nominal input. For the case of four inputs and five basis functions, the nominal output,  $y_0$ , and the perturbed response,  $y_i^l$ , are computed using the following inputs:

$$\begin{bmatrix} u_0^1 & u_0^2 & u_0^3 & u_0^4 \end{bmatrix}^T \implies y_0 \quad (4.6)$$

$$\begin{bmatrix} u_0^1 & u_3^2 & u_0^3 & u_0^4 \end{bmatrix}^T \implies y_3^2 \quad (4.7)$$

where we have used the response to the second input perturbed by the third basis function as an example for  $y_i^l$ . Next, we generate the perturbational component of the response by

$$\delta \underline{y}_i^l = y_i^l - y_0 \quad (4.8)$$

and, given the assumption that  $\delta \underline{y} = Y \underline{\eta}$ , we may finally form the output relationship matrix. Similar to the  $U$  matrix, the  $Y$  matrix involves a peculiar dimensionality, such that the dimensions of the output relationship matrix for a system with  $n$  outputs,  $m$  inputs,  $n_{basis}$  basis functions, and a discrete-time horizon of length  $N$  are  $n \times (m * n_{basis}) \times N$ . The column vector  $\delta \underline{y}_i^l$ , of dimension  $n \times 1 \times N$ , then becomes column  $(l - 1) * n_{basis} + i$  in the  $Y$  matrix.

Now that we have generated the  $U$  and  $Y$  matrices, we are ready to address the optimization routine. The minimization of the cost defined by Equation (2.33) is handled by *quadprog*, a quadratic programming function available in MATLAB<sup>®</sup>, which solves



constrained quadratic minimization problems of the form given below

$$\min_{\underline{x}} (0.5 * \underline{x}^T \Lambda \underline{x} + \underline{v}^T \underline{x}), \text{ such that } A \underline{x} \leq \underline{b} \quad (4.9)$$

However, given the unique dimensions of the  $U$  and  $Y$  matrices, a different manner of expressing the cost function is necessary. We first rewrite the cost function of Equation (2.33) in the proper summation notation as

$$J_k = \sum_{i=k}^{k+N-1} \{ \underline{\eta}^T [Y^T(i) \Gamma Y(i) + U^T(i) \Upsilon U(i)] \underline{\eta} + 2[(\underline{y}_0(i) - \underline{r}(i))^T \Gamma Y(i) + \underline{u}_0^T(i) \Upsilon U(i)] \underline{\eta} \} \quad (4.10)$$

where the notation  $Y(i)$  refers to the value of the  $Y$  matrix at time sample  $i$  such that  $Y(i) = Y(:, :, i)$ , with similar definitions for the  $U$  matrix and  $\underline{y}_0$ ,  $\underline{r}$ , and  $\underline{u}_0$  vectors. In order to cast this cost function in the form of Equation (4.9), we note that the summation may be collapsed into a single expression by summing the appropriate terms over their time dimension such that we obtain

$$\Lambda_k = \sum_{i=k}^{k+N-1} [Y^T(i) \Gamma Y(i) + U^T(i) \Upsilon U(i)] \quad (4.11)$$

$$\underline{v}_k^T = \sum_{i=k}^{k+N-1} [(\underline{y}_0(i) - \underline{r}(i))^T \Gamma Y(i) + \underline{u}_0^T(i) \Upsilon U(i)] \quad (4.12)$$

$$J_k = \underline{\eta}^T \Lambda_k \underline{\eta} + 2 \underline{v}_k^T \underline{\eta} \quad (4.13)$$

The cost function,  $J_k$ , is now in the format required by *quadprog*.

However, when we attempt to implement this command generation technique in MATLAB<sup>®</sup> software, we encounter a major problem. The perturbation input, obtained by

$$\delta \underline{u}(i) = U(i) \underline{\eta} \quad (4.14)$$

where  $\delta \underline{u}(i)$  is the perturbational input at time sample  $i$  and  $U(i)$  is the value of the  $U$  matrix at time sample  $i$ , tends to generate illogical control inputs. For example, most of the test cases used during the development of this command generator resulted in perturbational inputs for the aileron, elevator, and rudder deflection angles on the order of

$\pm 3600^\circ$ . Clearly an input of this magnitude is not a rational option to apply to a control surface which has physical deflection angle limits on the order of  $\pm 30^\circ$ . Nevertheless, the simulation attempts to apply these control inputs and, since the script-file which represents the air vehicle limits the control inputs based on internal saturation values, a system response is obtained for a short period of time until the operation of the script-file fails. The irrational input values appear to be caused by an error in the method used to compute the  $\Lambda$  matrix and the  $\underline{v}^T$  vector. Once this computational method was corrected, the perturbational inputs took on more realistic values and the simulation would persist for a longer period of time. Yet, even when we attempt to fly the munition at a constant speed in a steady-level configuration, the script-file which represents the vehicle's dynamics numerically continues to fail in a similar manner. Given that previous research efforts [9, 15, 16] have achieved successful MPC optimization routines in two dimensions, this problem appears to be caused by the details of our MATLAB<sup>®</sup> implementation of the six-degree-of-freedom dynamics model.

*4.2.2 MEP-based Command Generator.* In Section 3.5.3 we presented a modified minimum effort control scheme that was based on the information provided by the waypoints which constitute the minimum effort path. This MEP-based command generation process is decomposed into two separate subfunctions. One routine determines when the current waypoint has been reached and then commands the munition to fly to the next waypoint along the pre-planned path. The other subfunction generates the actual control inputs in a manner which produces smooth command sequences. The following discussion presents the details of the MATLAB<sup>®</sup> implementation of the MEP-based command generator.

First, before any control inputs are computed, we must formulate a method to declare a waypoint has been reached. Given the near certainty that the actual flight path will not pass directly through the waypoint, we are forced to make this declaration based on the munition's proximity to the current waypoint. One simple solution to this problem is to make the decision to fly to the next waypoint when the actual trajectory passes within a specific distance of the current waypoint. However, there is still no guarantee that

the munition's trajectory will satisfy this proximity criterion. A more robust solution to this issue was achieved by monitoring the trend of the distance between the munition and the current waypoint. Since we expect this distance to be a monotonically decreasing function of time, the decision to proceed to the next waypoint is made when the distance at the current time sample is greater than the distance at the previous time sample. In order to provide an extra level of confidence, this decision may be easily delayed until the increasing distance phenomenon is observed over a few consecutive time epochs. Once the decision is made, due to the nonlinear nature of this six-degree-of-freedom model, we must recompute the trim conditions for the air vehicle to fly at the new heading angle and flight-path angle, as defined by the vector from the munition's current position to the redefined current waypoint.

As stated in Section 3.5.3, the control inputs are obtained directly from the position data which define the current and previous waypoints. Furthermore, we compute the control inputs such that we avoid large-magnitude step changes at the inputs. These smooth-transitioning command sequences are necessary, since the simulation has a tendency to crash when a large-magnitude change to the input is commanded within a single time sample. Therefore, a simple command generation technique was used to generate the following input vector:

$$\underline{u} = \begin{bmatrix} V_{cmd} \\ h_{cmd} \\ \Psi_{cmd} \end{bmatrix} \quad (4.15)$$

where  $V_{cmd}$  is the desired constant speed,  $h_{cmd}$  is the desired altitude, and  $\Psi_{cmd}$  is the desired heading angle. In order to smooth out the  $h_{cmd}$  input, we use the munition's altitude at the time when the previous waypoint was reached as the starting point for a straight-line segment which terminates at the altitude of the current waypoint. The commands at each time sample for this straight-line segment are generated as a simple time propagation at the desired speed along the flight-path angle determined by a LOS vector between the previous and current waypoints. The exact magnitude of the incremental change in altitude is determined by the sample interval, vehicle speed, and flight-path

Table 4.1 Starting location and target position for basic demonstration

	North (ft)	East (ft)	Altitude (ft)
Munition Start	0	0	1000
Target Location	20000	0	0

angle

$$\Delta h_{cmd} = T_s V_{cmd} \sin(\gamma_{path}) \quad (4.16)$$

For the transitions on the  $\Psi_{cmd}$  input, we employ a static turn rate value, chosen in an *ad hoc* manner, to compute small-magnitude incremental changes to the input at each time sample. For example, in a discrete-time simulation with a time step of 0.01 s and a static turn rate of 30 degrees per second, the MEP-based command generator computes incremental heading changes of, at most,  $0.3^\circ$  at each time sample. We are now ready to present the simulation scenarios, starting with a basic demonstration of the simulation environment.

### 4.3 Basic Demonstration

Before we address the various test cases, we first demonstrate the simulation environment through two basic scenarios in which we can predict the selected path. In these two examples, we place obstacles in a manner that forces the minimum-distance path and the minimum-effort path to be identical. These examples will also serve the purpose of demonstrating the typical situations in which other researchers [9] have used model predictive control in two dimensions. Even though we have not been able to replicate this MPC command generation technique, we have at least formulated a method to produce the waypoints and reference trajectories required by the MPC approach. For simplicity, both examples will use the same initial and termination positions as given in Table 4.1.

In the first situation, we construct the scene by placing a short, wide obstacle directly in between the munition and the target in order to force the path planner to choose the up-and-over path. This simple scene is shown with all the candidate paths in Figure 4.1 where the red path is the chosen MDP/MEP candidate, and the blue paths are the other non-selected candidate paths. Given the definition of the path selection cost function in

Table 4.2 Path selection cost values for scenario in which trajectory is forced up-and-over the obstacle - Path 1 is up-and-over, Paths 2 and 3 are around, Paths 4 and 5 are up-and-around

	Path #1	Path #2	Path #3	Path #4	Path #5
Orientation	0.208203	0.00143559	0.001436	0.394462	0.394462
Travel	0.200385	0.198368	0.198368	0.201438	0.201438
Distance	0.07008	0.370164	0.370164	0.094796	0.094796
Total Cost	<b>0.478668</b>	0.569968	0.569968	0.690696	0.690696

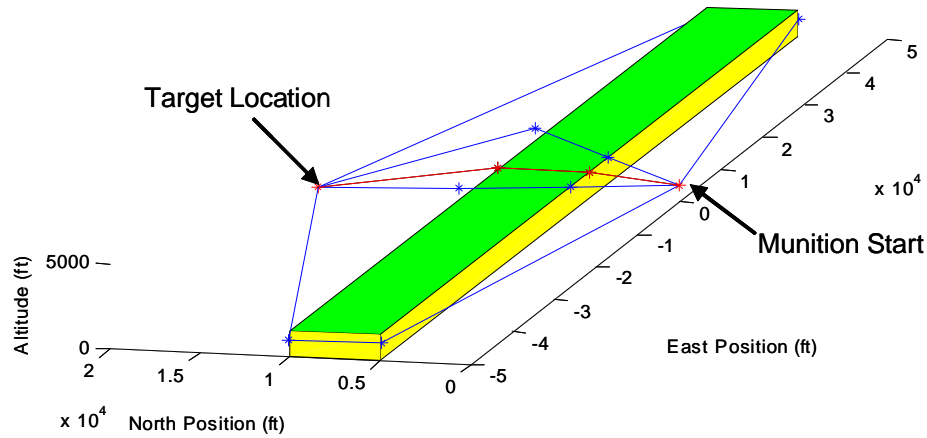


Figure 4.1 Scenario in which trajectory was forced up-and-over the obstacle - red path is MEP/MDP and blue paths are non-selected candidates

Equations (3.6) and (3.22), we expect the around paths to have large values for  $J_{distance}$  and  $J_{travel}$ . Furthermore, knowing that the up-and-over path will be the minimum-distance path, we anticipate this path will have the smallest value for  $J_{distance}$ . The individual cost values for each path are shown in Table 4.2 and the trend for  $J_{distance}$  matches our expectation.

The second example is generated by using a tall, thin obstacle in an effort to force the path planner to choose the around path. This scenario is depicted in Figure 4.2 where the red path is the chosen MDP/MEP candidate, the magenta paths are candidate paths with unstable systems, and the blue paths are the remaining non-selected candidate paths. A discussion of these unstable paths is presented in Section 4.4.2. Similar to the previous example, we expect the around paths to have the minimum value for  $J_{distance}$ . Since the

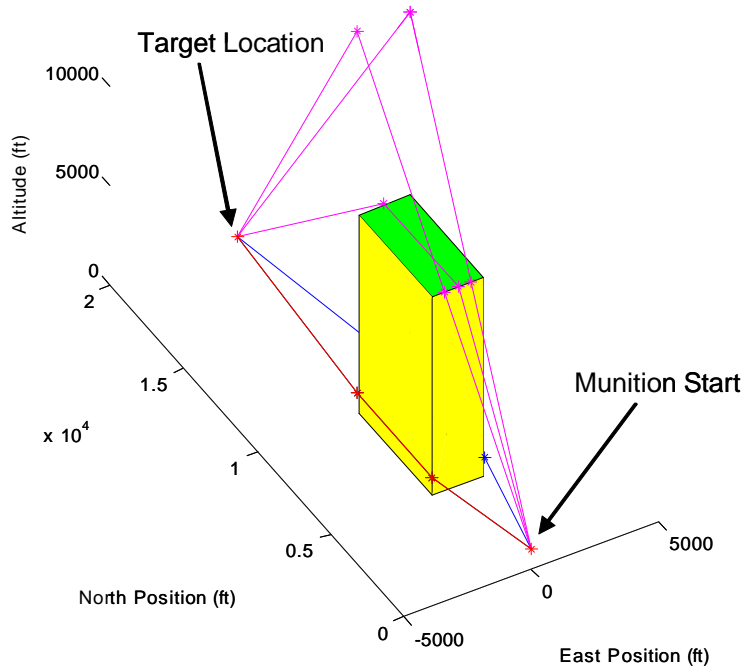


Figure 4.2 Scenario where the selection of the around path is forced - red path is MEP/MDP, magenta paths are unstable/infeasible, and blue path is an unselected candidate

up-and-over and up-and-around paths produce unstable systems during the initial climb to the top of the obstacle, these three candidate paths must be ignored and are given component and total costs equal to zero. Table 4.3 shows the component and total costs for each candidate path in this example.

#### 4.4 Test Set 1: Path Planning Characteristics

*4.4.1 Computational Complexity.* One of the major shortcomings of this minimum effort control based approach is the computational complexity of the path planning algorithm. During the development of the waypoint generation procedure in Section 3.3, it was observed that the maximum number of potential paths grows exponentially based on the number of obstacles blocking the LOS path from the munition to the target. Figure

Table 4.3 Path selection cost values for scenario in which trajectory is forced around the obstacle - Path 1 is up-and-over, Paths 2 and 3 are around, Paths 4 and 5 are up-and-around

	Path #1	Path #2	Path #3	Path #4	Path #5
Orientation	0	0.499999	0.500001	0	0
Travel	0	0.5	0.5	0	0
Distance	0	0.5	0.5	0	0
Total Cost	0	<b>1.499999</b>	1.500001	0	0

Table 4.4 Path complexity data

	$t_{wp\_gen}$	$t_{path\_sel}$	$n_{paths}$
Single-Obstacle Case	0.32 s	31.586 s	5
Two-Obstacle Case	5.138 s	235.944 s	25
Three-Obstacle Case	19.328 s	1622.724 s	117

4.3 depicts the evolution of candidate paths when the munition must avoid an increasing number of obstacles.

In Table 4.4, we can clearly see the rapid growth in the number of candidate paths,  $n_{paths}$ , that are passed to the path selection routine. Another indication of the computational complexity inherent to this path planning algorithm is the amount of time spent generating waypoints,  $t_{wp\_gen}$ , and selecting the minimum effort path,  $t_{path\_sel}$ . Since the minimum effort path planning algorithm used in these simulations was not implemented with any deliberate attempt at efficiency, the exact amount of time required by these tasks is not of interest. However, the key point to notice is the exponential growth in the amount of time necessary for the waypoint generation and path selection functions to complete their tasks as shown in Figure 4.4. Notice that the number of candidate paths grows by a factor of 5 for each obstacle that must be avoided, while the associated path selection time grows by a factor of 7. Methods to mitigate this exponential growth are a subject of future research.

*4.4.2 Unstable Paths.* The primary advantage of this path planning technique is the built-in capability to guide the munition along a path that is assured to be feasible, in terms of system stability and controllability, while the minimum distance approach to path planning cannot make such an assurance. There is no guarantee that the minimum

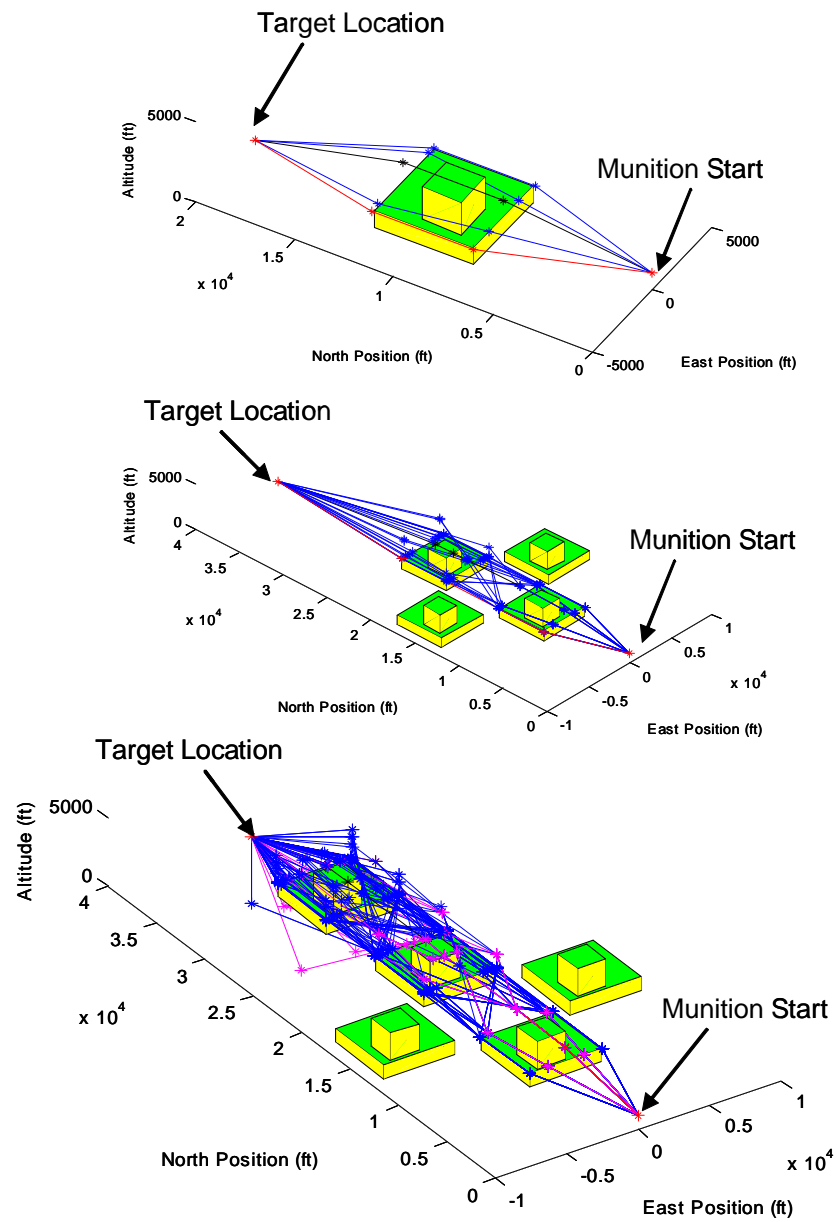


Figure 4.3 Increasing path complexity - red path is MEP and black path is MDP



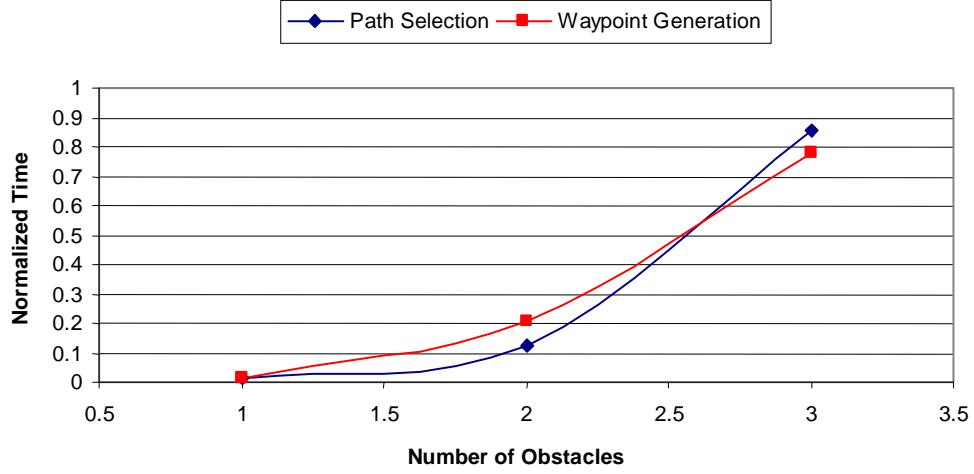


Figure 4.4 Trend data for growth of waypoint generation and path selection times

distance path will lead to a stable system throughout the munition's flight. One such instance is created by modifying the scenario depicted in Figure 4.3 and moving the target's position closer to the obstacle. Graphically, this is shown in Figure 4.5.

In order to demonstrate the unstable system, we bypass the path selection function and force the munition to traverse the minimum distance path. The result of flying the munition along this path is shown in Figure 4.6 and, by zooming in on the boxed area, Figure 4.7 shows that the system quickly becomes unstable after reaching the waypoint on the far side of the obstacle. One of the main benefits of the minimum effort path planning algorithm is exemplified by this test case. The MEP algorithm has the inherent ability to predict the emergence of system instabilities on each of the potential paths and then ignore these unstable paths. This prediction capability is realized during the controllability Grammian computation of the path selection process, wherein we compute a linearized model of the closed-loop system. In order to anticipate an unstable system, we investigate the eigenvalues of the closed-loop system dynamics matrix and, using classical stability definitions [14], declare the system to be unstable if any of the eigenvalues are greater than zero.

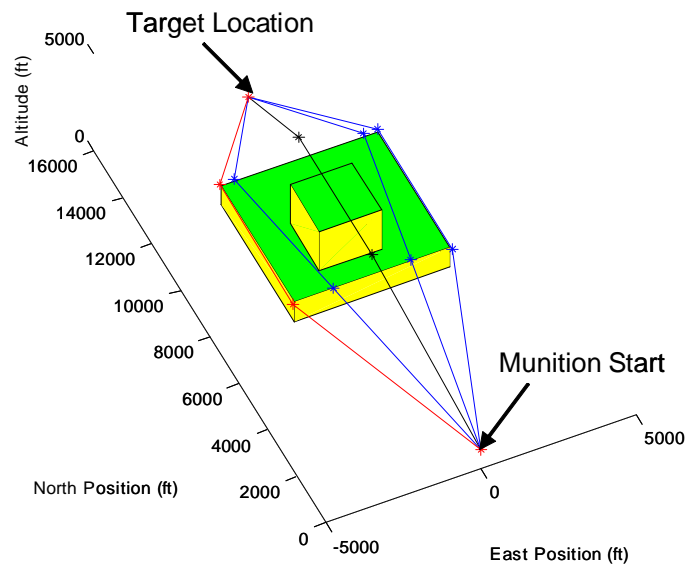


Figure 4.5 Scenario where MDP produces unstable system

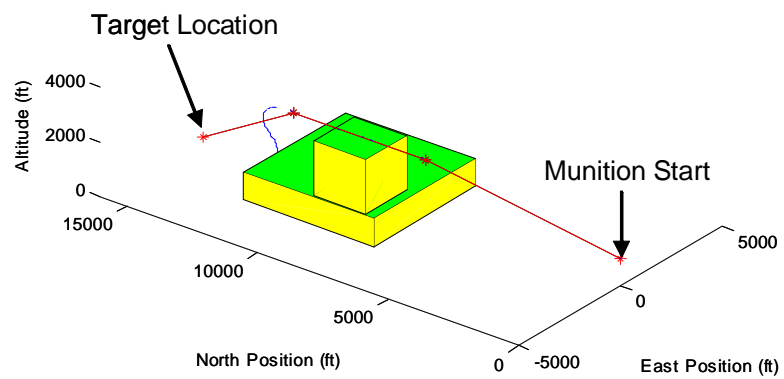


Figure 4.6 Unstable system response using MDP

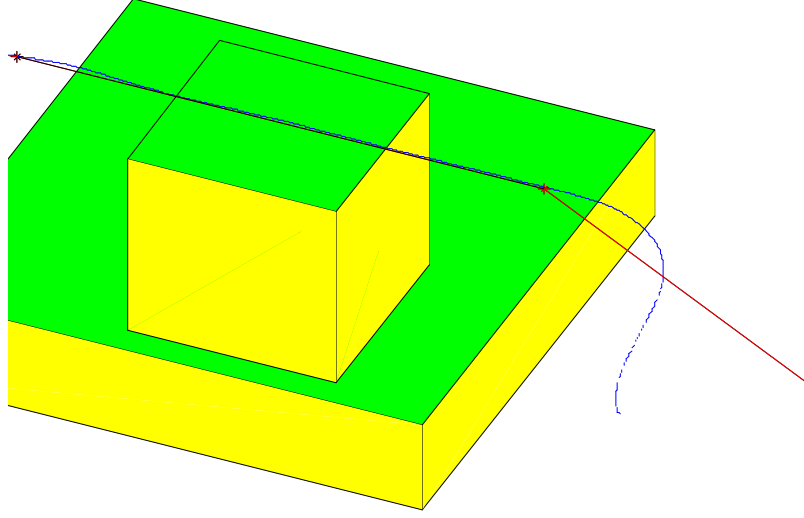


Figure 4.7 Zoomed in view of unstable system response using MDP

#### 4.5 Test Set 2: MEP-based Control Characteristics

*4.5.1 Target Location Relative to Obstacles.* One situation in which the minimum effort control approach to input generation is highly effective is when the obstacles and the target are sufficiently separated. An example of this case is shown in Figure 4.8 where the red path is the MEP, the black path is the MDP, and the blue path is the actual flight trajectory. Notice that the flight trajectory misses all of the obstacles and terminates at the target (within a given threshold distance). This small threshold value is necessary since the target has its own size characteristics and is not simply a point-mass target. Through this scenario, we see that the minimum effort control technique is capable of generating control inputs that permit the munition to impact the target. This is only possible when the munition has enough time to complete the heading changes after reaching each waypoint.

For the case of a target located near an obstacle, the minimum effort control technique for command generation performs very poorly. An example of this case is shown in Figure 4.9 where the red path is the MEP, the black path is the MDP, and the blue path is the actual flight trajectory. Even though this control algorithm does not drive the munition into any of the obstacles, minimum effort control fails to fly the munition to the target

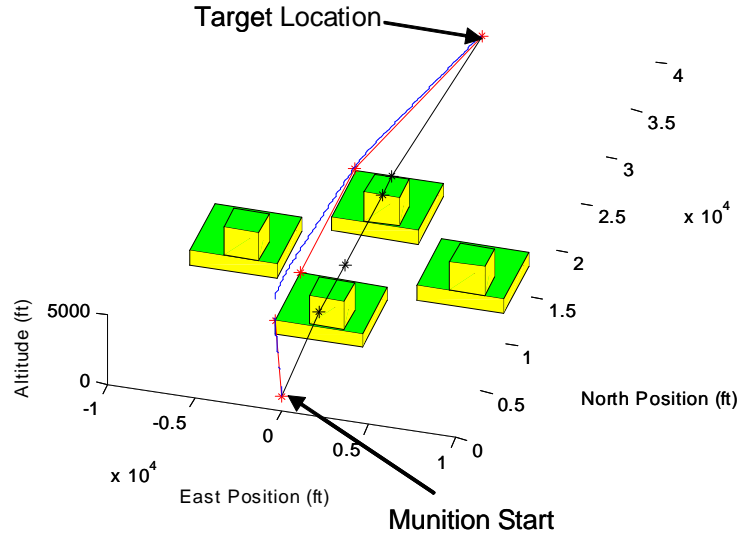


Figure 4.8 Target in the open

since there is not enough space and time to complete the turn before the munition passes the target. Note that in this case, the MDP solution is not a viable alternative since it would produce an unstable system in a manner similar to the scenario of Section 4.4.2.

*4.5.2 Narrow Alley Scenario.* A severe downside to the minimum effort control command generation method is that it does not force the munition back onto the LOS trajectory after a heading change is completed. While this is not a major cause for concern in situations where the obstacles are spaced far apart, as the separation between obstacles shrinks, the likelihood of the munition impacting an obstacle during and after turns continually increases.

An example of this narrow alley scenario is depicted in Figure 4.10 where the MEP solution (red path) instructs the munition to travel through the small corridor between two obstacles. In order to better illustrate this problem, we zoom in on the boxed region. This enlarged view is presented in Figure 4.11 where we can clearly see the actual flight trajectory (the blue path) passing through the obstacle. Again, this is due to the fact that the munition cannot execute instantaneous turns and minimum effort control commanding

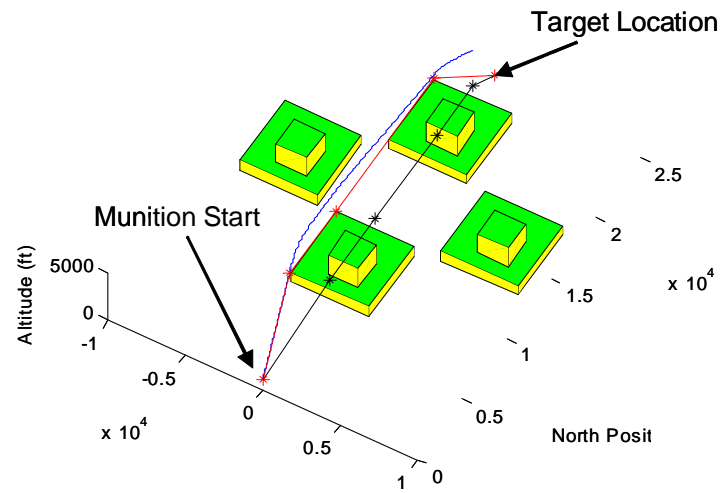


Figure 4.9 Target located close to obstacle

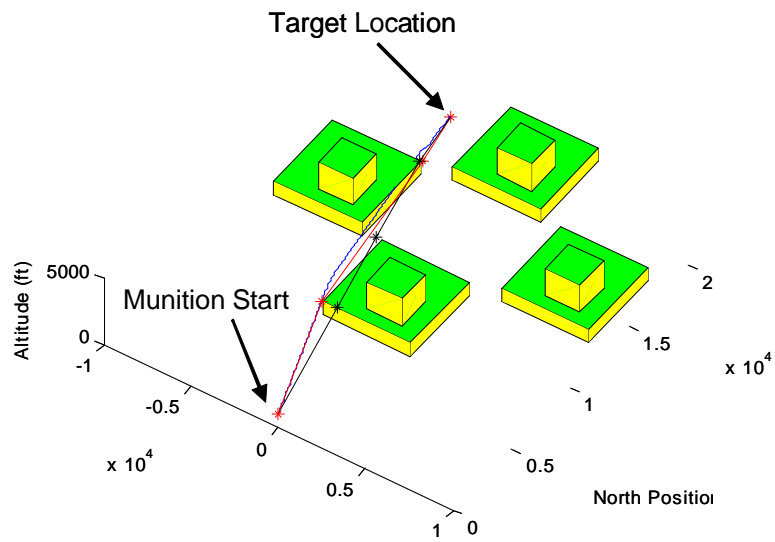


Figure 4.10 Narrow alley scenario

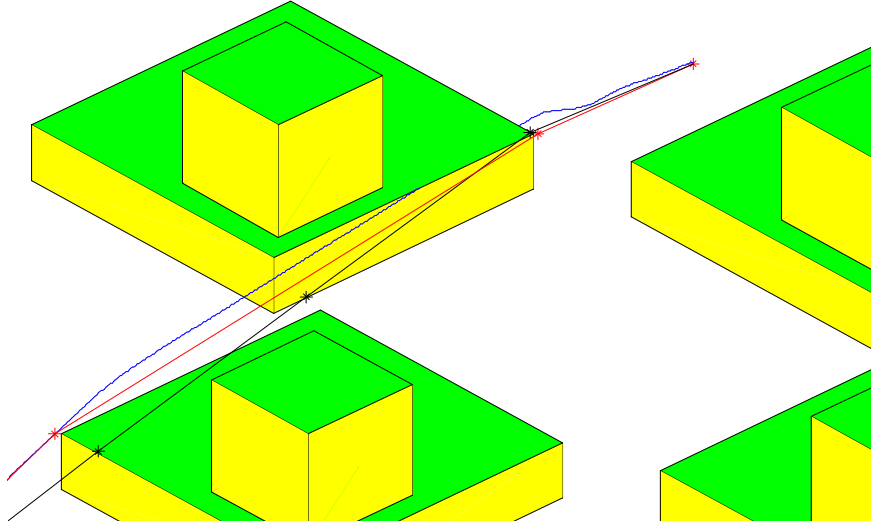


Figure 4.11 Zoomed in view of narrow alley scenario

does not guide the munition back to the unobstructed line-of-sight path between waypoints. The development of alternative command generators is a subject for future research.

*4.5.3 Target Designation Change Scenario.* A case of great interest for the multi-target situation is a demonstration of the minimum effort control method in response to a change in the target designation during the munition's flight. An example of this situation is shown in Figure 4.12 where the red path is the final MEP, the blue path is the actual flight trajectory, and the magenta path is the initial MEP. The starting location for the munition and both target positions are given in Table 4.5. We can clearly see that the path planning algorithm accomplishes its task of generating an updated MEP in response to the change in the termination point. However, the control input generation scheme must now adapt to this newly planned path and fly the munition toward a new target point. The control algorithm is just as capable of completing this task as it would be if the scenario had started with this new target. We still need to be concerned with the potential for impacting an obstacle as presented in the narrow-alley scenario and the amount of space required to complete a turn as shown in the case of a target located close to an obstacle. Overall, the minimum effort path planning algorithm is adaptable to target reassignment.

Table 4.5 Starting location for munition and target positions for target designation change scenario

	North (ft)	East (ft)	Altitude (ft)
Munition Start	0	0	1000
Initial Target	30000	0	0
Updated Target	24000	-7500	0

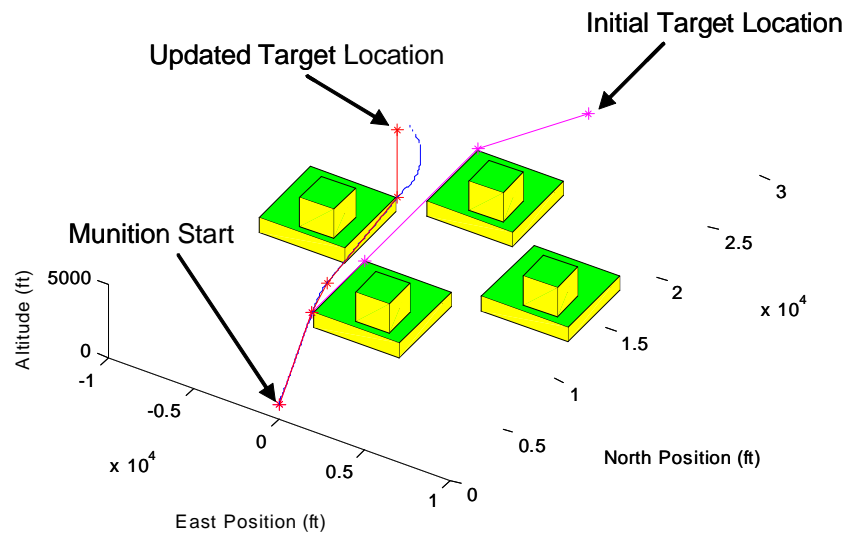


Figure 4.12 Target designation change scenario

#### 4.6 Summary

In this chapter, we described the implementation of the command generators in MATLAB<sup>®</sup> and analyzed the results of various test scenarios. Specifically, Section 4.2 detailed the implementation of the MPC and MEP-based command generation techniques and discussed the major issues which precluded the MPC approach from being employed in the simulations. Next, Section 4.3 presented a simple demonstration of the simulation environment in which the scene was constructed such that the munition is forced to select either the up-and-over or around paths. Next, Section 4.4 investigated the characteristics of the MEP path planning algorithm, highlighting the rapid growth in computational complexity and the inherent ability of this algorithm to predict the emergence of system instabilities. Finally, Section 4.5 tested the capability of the MEP-based command generation approach in various situations of interest, to include the case of a large change to the target's location while the munition is in mid-flight. We are now ready to conclude this research and provide recommendations for future research efforts on this topic.



## *V. Conclusions and Recommendations*

### *5.1 Research Goal*

As stated in Chapter 1, the goal of this research was to develop a path planning algorithm that provides an obstacle avoidance function in 3-dimensional space and to investigate the application of model predictive control, circle-line-circle control, and minimum effort control optimization techniques in this 3-dimensional setting. Finally, the overall guidance and control scheme must operate in an environment in which the designated target is subject to change.

### *5.2 Conclusions*

The path planning algorithm developed by this research was segmented into two functions. First, the waypoint generation routine generates sets of waypoints that define candidate paths in a manner that guarantees obstacle avoidance along the line-of-sight path between successive waypoints. Next, a path selection procedure, based on the concept of controllability Grammians, chose the candidate path which requires the least amount of control input energy to traverse. This path planning method has proved itself to be effective at producing guidance paths that are free of obstructions. Additionally, the path chosen by this algorithm is also guaranteed to be feasible. This assurance is possible since the controllability Grammian approach to path selection facilitates the identification of unstable systems and the detection of attempts to move the system in an uncontrollable direction of the state-space. However, there is a large overhead cost in terms of computational time required by this method, and this cost increases exponentially with the number of obstacles that must be avoided. One significant shortcoming of this path planning algorithm, though it appears to be caused by the implementation method, is that it has a strong bias towards flying at lower altitudes, making the selection of the up-and-over path a very rare occurrence. A suggested method for dealing with this problem is presented in Section 5.3.4.

In Chapter 3, we attempted to extend the MPC and CLC optimization techniques to 3-dimensional problems. While both of these efforts did not achieve the intended level of success, we have gained insight into how the extension may be completed.

For the model predictive control optimization scheme, we completed the theoretical extension of the problem in [9, 15] to three dimensions but encountered numerous failures during the implementation of this command generation method. Two efforts were necessary in order to accomplish the extension of MPC to 3-dimensions. First, the dimensions of the matrices and vectors required by the quadratic programming optimization technique had to be greatly increased in order to accommodate the larger dimensions of the air vehicle's system model. As a complement to this first modification, the reference trajectory had to be specified in three dimensions in order to take full advantage of this optimization method. Unfortunately, the implementation of this MPC approach in MATLAB<sup>®</sup>, as shown in Section 4.2.1, never produced meaningful results. Nevertheless, once we achieve a functioning version of MPC in two dimensions, the 3-dimensional implementation will be realized by simply replacing the 2-dimensional system with the flight model and reference path defined in three dimensions.

In the case of the extremal control law of the CLC approach, we find that an additional relationship is required in order to facilitate a trade-off between turn rate and pitch rate. A simple maximum turn rate constraint was sufficient in the 2-dimensional problem of [1] because it was assumed that the air vehicle maintained a specific altitude throughout its flight. When we relax this assumption and allow the vehicle to be free to move in the full 3-dimensional space, we obviously need to place a constraint on the maximum pitch rate as well as the turn rate. Moreover, given that the vehicle's three axes of motion are not decoupled, a third constraint must be invoked to describe the coupling between turn rate and pitch rate.

Finally, the results of applying input commands generated by the minimum effort control technique were discussed in Section 4.5. Through simulated target engagement scenarios, we have determined that this input generation method is an inadequate option. In general, the minimum effort control technique is not useful for this class of problem, since it does not have the ability to follow a desired path between waypoints. The negative

impact of this limitation varies based on the setup of the scenario under consideration. For cases in which the target is located close to a large heading change, we simply fly past the target before the turn is completed, but at least the munition is able to continue to engage targets. In the more severe case of a guidance path which attempts to fly the munition along a narrow corridor between obstacles, we may actually fail to achieve the obstacle avoidance requirement and fly the munition into the side of a mountain.

### 5.3 Recommendations

Given the successes and failures of this research, several recommendations can be made for future efforts on this topic.

*5.3.1 CLC Optimization in 3 Dimensions.* As stated in Section 3.5.1, the extension of the CLC (extremal) control optimization technique is incomplete. In order to finish this extension, a relationship must be established between turn rate and pitch rate. The requirement for this relationship is motivated by the fact that motion in the horizontal and vertical planes of the air vehicle are not fully decoupled. For example, when the munition attempts to execute a horizontal coordinated turn, the air vehicle is forced into a dive unless an intentional effort is made to counteract this effect. While it is known that this relationship exists, the exact (or even an approximate) form of the relationship is unknown. Once this trade-off between turn rate and pitch rate is characterized, the extension of the CLC control optimization technique may be completed in a manner similar to [1].

*5.3.2 MPC Implementation.* As shown throughout this research, the MATLAB<sup>®</sup> implementation of the MPC optimization technique is flawed, such that the simulation cannot even complete a flight along steady-level path. The notion that the flaw resides within this specific implementation is supported by the fact that other researchers [9, 15] have been successful in using the MPC technique. Bearing this in mind, it would be beneficial to work on coding a 2-dimensional problem properly and then expand the utility of this technique to 3-dimensional problems via the method presented in this thesis.

*5.3.3 Enhanced Waypoint Generation Technique.* The waypoint generation approach utilized throughout this research is a simple method designed to facilitate the investigation of the minimum effort path selection technique. A desired enhancement for the waypoint generation function is to allow for a terrain-following feature so that the vehicle may remain close to the ground and avoid detection. In order to realize this feature, we would seek to fly at a low altitude as often as possible and, when we are forced to a higher altitude, to execute climbing maneuvers at the maximum reasonable flight-path angle. This improvement would also require an alteration to the relative weighting values in the cost computation.

*5.3.4 CLC Assistance to Waypoint Generator.* The CLC command generation method is a very intuitive approach and, once the extension to 3 dimensions is complete, elements of the CLC optimizations technique may be used to augment the quality of the path planning process. Specifically, the details of the  $\kappa$ -trajectories dictate where the waypoint can be placed, relative to the corner of the obstacle, in order to ensure that the munition's actual trajectory does not impact an obstacle when the control inputs drive the trajectory away from the LOS path during heading changes.

*5.3.5 Remove Altitude Bias from Energy Computation.* The structure of the linearized model used for the controllability Grammian computation leads to an inherent bias towards flying at low altitudes. This happens because the definition of the control input vector contains an element for absolute altitude and a squared altitude value will become involved in any control input energy computation. Additionally, the control input vector composed of elements for speed, altitude, and heading does not provide an accurate representation of the control energy since these three inputs are actually pseudo inputs. A more logical choice, would be to minimize the control energy for the control input vector (throttle setting and 3 actuator deflection angles) that is applied to the non-linear air vehicle model. This change can be achieved by altering the structure of the model's input control vector and redesigning the controller to meet stability and performance requirements.

*5.3.6 Situational Logic for Target Designation Change Scenario.* In the target designation change scenario, we noted that the path planning algorithm was able to react to the updated target location. However, as shown in Figure 4.12, the munition is susceptible to flying past the target. For this research, we designated this situation as a failed engagement and halted the simulation. A desired alternative to this approach is to allow the munition to either search for a new target or to attempt a "second pass" against the current target. These two alternative options may be implemented by using situation-based logic to determine the appropriate course of action and, if necessary, a modified version of the path planning algorithm to determine the reference trajectory for a "second pass" engagement.

*5.3.7 Streamline Simulation Code for Efficiency.* Finally, an effort needs to be made to clean up the MATLAB<sup>®</sup> code implementation in order to streamline the path planning algorithm. One major step is to store the candidate paths in a branching tree structure, instead of a growing length list, so that we are able to avoid redundant computations when generating the path cost for the path selection process. Another major focus area is to prune off candidate paths earlier in the algorithm in an effort to reduce the number of unrealistic paths which must be maintained and evaluated throughout the simulation. These two courses of action could significantly reduce the computation complexity of the path selection algorithm.

## Bibliography

1. Anderson, Erik P. *Extremal Control and Unmanned Air Vehicle Trajectory Generation*. MS thesis, Brigham Young University, Provo, UT, April 2002.
2. Anderson, Erik P. and Randal W. Beard. "An Algorithmic Implementation of Constrained Extremal Control for UAVs." *AIAA Guidance, Navigation, and Control Conference and Exhibit*. August 2002.
3. Anderson, Erik P., Randal W. Beard and Timothy W. McLain. "Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 3, 471–477 (May 2005).
4. Bar-Shalom, Yaakov, X. Rong Li and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software*. New York, NY: John Wiley and Sons, Inc., 2001.
5. Beard, Randal W., Timothy W. McLain Michael A. Goodrich and Erik P. Anderson. "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 6, 911–922 (December 2002).
6. Blackman, Samuel and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.
7. Gajic, Zoran and Muhidin Lelic. *Modern Control Systems Engineering*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1996.
8. Howlett, Jason K. *Path Planning for Sensing Multiple Targets from an Aircraft*. MS thesis, Brigham Young University, Provo, UT, April 2003.
9. Lapp, Tiffany and Leena Singh. "Model Predictive Control Based Trajectory Optimization for Nap-of-the-Earth (NOE) Flight Including Obstacle Avoidance." *Proceedings of the 2004 American Control Conference*. 891–896. 30 June - 2 July 2004.
10. Li, X. Rong and Vesselin P. Jilkov. "Survey of Maneuvering Target Tracking Part I: Dynamic Models," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 39, No. 4, 1333–1364 (October 2003).
11. Liebst, B.S. and C.H. Spenny. "Nonlinear Dynamic Model of the F-16 Aircraft." Unpublished user's manual, December 2000.
12. Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, Vol. 1. Arlington, VA: Navtech Book and Software Store, 1994.
13. Mayne, David Q. and Hannah Michalska. "Receding Horizon Control of Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. 35, No. 7, 814–824 (July 1990).
14. Reid, J. Gary. *Linear Systems Fundamentals: Continuous and Discrete, Classic and Modern*. New York, NY: McGraw-Hill, Inc., 1983.

15. Singh, Leena and James Fuller. "Trajectory Generation for a UAV in Urban Terrain, Using Nonlinear MPC." *Proceedings of the 2001 American Control Conference*. 2301–2308. June 2001.
16. Singh, Leena and Tiffany Lapp. "Model Predictive Control in Nap-of-Earth Flight Using Polynomial Control Basis Functions." *Proceedings of the 2005 American Control Conference*. 840–845. June 2005.
17. Stevens, Brian L. and Frank L. Lewis. *Aircraft Control and Simulation* (2nd Edition). Hoboken, NJ: John Wiley and Sons, Inc., 2003.
18. Strang, Gilbert. *Linear Algebra and Its Applications* (3rd Edition). San Diego, CA: Thomson Learning, Inc., 1988.
19. Vincent, Thomas L. and Walter J. Grantham. *Nonlinear and Optimal Control Systems*. New York, NY: John Wiley and Sons, Inc., 1997.
20. Yang, Guang and Vikram Kapila. "Optimal Path Planning for Unmanned Air Vehicles with Kinematic and Tactical Constraints." *Proceedings of the 41st IEEE Conference on Decision and Control*. 1301–1306. December 2002.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
<b>1. REPORT DATE (DD-MM-YYYY)</b> 23-03-2006		<b>2. REPORT TYPE</b> Master's Thesis		<b>3. DATES COVERED (From - To)</b> Aug 2004-Mar 2006	
<b>4. TITLE AND SUBTITLE</b>  A Minimum Effort Control Approach to Guided Munition Path Planning				<b>5a. CONTRACT NUMBER</b>	
				<b>5b. GRANT NUMBER</b>	
				<b>5c. PROGRAM ELEMENT NUMBER</b>	
<b>6. AUTHOR(S)</b>  Borkowski, Jeffrey M., First Lieutenant, USAF				<b>5d. PROJECT NUMBER</b>	
				<b>5e. TASK NUMBER</b>	
				<b>5f. WORK UNIT NUMBER</b>	
<b>7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)</b> Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  AFIT/GE/ENG/06-07	
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> AFRL/MNGN ATTN: Mr. Johnny H. Evers 101 W. Eglin Blvd., Suite 212 Eglin AFB, FL 32542-6810 Comm: 850-882-2961 ext 2347 Email: johnny.evers@eglin.af.mil				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>	
				<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>	
<b>12. DISTRIBUTION/AVAILABILITY STATEMENT</b> APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
<b>13. SUPPLEMENTARY NOTES</b>					
<b>14. ABSTRACT</b> <p>An advance in the development of smart munitions entails autonomously modifying target selection during flight in order to maximize the value of the target being destroyed. Target identification and classification provides a basis for target value which is used in conjunction with multi-target tracks to determine an optimal aimpoint for the munition. A unique guidance law can be constructed that exploits attribute and kinematic data from an onboard video sensor. This thesis develops an innovative path planning algorithm that provides an obstacle avoidance function while navigating the munition toward the highest value target. The foundation of this path planning method is found in the principles of minimum effort control optimization. Results demonstrate the ability of the path planning algorithm to determine a path for the munition to follow which is both stable and feasible.</p>					
<b>15. SUBJECT TERMS</b> <p>Path planning, guidance, optimal control, minimum effort control, model predictive control, extremal control</p>					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>  UU	<b>18. NUMBER OF PAGES</b>  96	<b>19a. NAME OF RESPONSIBLE PERSON</b> Juan R. Vasquez, Lt Col (ENG)
REPORT U	ABSTRACT U	c. THIS PAGE U			<b>19b. TELEPHONE NUMBER (Include area code)</b> (937) 255-3636 ext 7231; email: juan.vasquez@afit.edu

**Standard Form 298 (Rev. 8-98)**  
Prescribed by ANSI Std. Z39-18